AN INTERACTION FRAMEWORK FOR MANAGING APPLICATIONS AND INPUT IN MULTIPLE DISPLAY ENVIRONMENTS

BY

JACOB T. BIEHL

B.S., University of Illinois at Urbana-Champaign, 2002
M.S., University of Illinois at Urbana-Champaign, 2004

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Associate Professor Brian P. Bailey, Chair
Assistant Professor Kyratso Karahalios
Professor Klara Nahrstedt
Associate Professor Michael Twidale
Professor Carl Gutwin, University of Saskatchewan

Abstract

In today's workplace, there is a continual increase in the size and complexity of the problems that knowledge workers are challenged to solve. This often requires workers with different expertise to come together to collaboratively develop solutions. To facilitate these collaborations, workers often turn to the use of digital information tools (electronic documents, web applications, analysis packages, etc.). While these tools support individual tasks well, existing system frameworks do not provide adequate support for their use in collaborative contexts, limiting their effectiveness. Researchers in the CSCW and HCI communities have proposed the use of multiple display environments (MDEs) as one potential solution for improving collaboration that hinges on the use of digital information.

This dissertation contributes the design, implementation, and evaluation of a new interaction framework, called IMPROMPTU, which enables users to more effectively collaborate using MDEs. Central to the framework's effectiveness is its flexible application sharing services that enable simultaneous, multi-user sharing of unmodified off-the-shelf applications. The framework's novelty is further reinforced through its negotiated interaction model, providing users a natural and intuitive process for establishing and managing shared applications. We also demonstrate the value and impact of IMPROMPTU in one of the first field studies to investigate the use of MDEs in authentic task domains. The innovation in the framework stemmed from leveraging and integrating knowledge from fundamental theories of collaboration, understanding existing collaborative practices related to the use of MDEs, and applying principled HCI design techniques. The culmination of this work moves MDE research out of the laboratory and into the real-world, informing the design of future frameworks and improving how users collaborate while utilizing digital information tools.

To Emily, Parker, and Jackson

Acknowledgments

The accomplishments in this dissertation would not have been possible without the insight, encouragement, and support of many people. I thank my advisor, Brian Bailey, for his guidance in helping me mature the ideas, methods, and results of this dissertation. Professor Bailey has not only shared with me his expertise in human-computer interaction research, but has also helped me find my own passion for conducting research.

I also thank Karrie Karahalios, Michael Twidale, Klara Nahrstedt, and Carl Gutwin for their willingness to serve on my committee and to provide valuable advice and feedback. I appreciate most of all their ability to provide a different perspective to my work.

William Baker deserves a lot of thanks for his help in implementing IMPROMPTU. He was instrumental in helping to tackle the large technical challenges that were faced in this dissertation. William is by far of the most talented people I have ever worked with.

I am very thankful for all the help and encouragement I received from Kori Inkpen, Desney Tan, and most of all, Mary Czerwinski at Microsoft Research. Mary, Kori, and Desney were instrumental in providing access and resources to study IMPROMPTU in the field. Without their support, the impact of this work would certainly be diminished. I am also thankful for the opportunity to intern at Microsoft Research and mature as a researcher. At MSR I was able to gain new perspectives in research methodologies, learn to be more independent and self-guided in my work, and become more creative in solving challenging problems.

My family deserves an immense amount of gratitude for supporting me through this endeavor. My wife, Emily, has given her patience, encouragement, and love – even when

I didn’t deserve it. I also thank my parents, Gary and Debbie, for always supporting me in my decisions and for always being proud of the things that I accomplish.

I will be forever grateful for the mentoring that Tom and Janet Eakman have provided over the years. They have taught through demonstration many important life and scholarly lessons. These include how to be tactful in difficult situations, the necessity of effective communication, and the importance of being thoughtful and thorough when making decisions. Most of all, Tom and Janet showed me the value of treating everyone as equals, while earning their respect along the way.

I would certainly not have survived graduate school without the support of my academic sister. Shamsi Iqbal has been the figurative and literal “shoulder to lean on” throughout graduate school. I am grateful for her wisdom, support, and most of all, willingness to listen. Over many cups of coffee, Shamsi and I have shared both our accomplishments and setbacks.

I am also lucky to have formed many rich friendships while at Illinois. These outstanding individuals include Adam Lee, Tanya Crenshaw, Erin Chambers, Ramona Thompson, Jodie Boyer, Eric Gilbert, Tony Bergstrom and Nila Sharmin. Through even more cups of coffee, conversation, and an *occasional* beer, these friends helped me keep a level head, see things from different perspectives, or just simply distracted me from work when I needed to be distracted.

Finally, I am infinitely thankful for all the assistance provided by Anda Ohlson, Kay Tomlin, Mary Beth Kelley, Barb Cicone, and Angie Bingaman. Whatever the issue, I knew I could count on them to find a solution.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Collaboration is an essential and fundamental activity of daily work. The average knowledge worker spends 37% of her day engaged in meetings with one or more individual; with 66% of these meetings taking place in face-to-face, co-located sessions [91]. Users benefit greatly from these collaborations. Individuals who work in groups are known to be better at addressing, decomposing, and developing solutions to problems [29]. Working in groups also often produces more, higher quality ideas compared to working individually [41, 101]. It can even improve trust and sense of belonging in a work place [50, 143].

To best support collaboration, workers often leverage the use of shared information artifacts to build common ground, compare alternative ideas, keep track of past discussions, capture solutions, and much more [28, 79, 125, 140]. In the modern workplace, there is a significant need to collaborate with *digital* information artifacts. For example, this need is manifested when groups work on a complex task where the final result will be in digital form (e.g. a presentation or digital document), where access to real time or constantly changing data is required (e.g. stock market data or electronic commerce), or where the task creates or requires access to multimedia content sources (e.g. news or surveillance video).

Figure 1.1: The scenario depicted in this photo highlights the current challenges of collaborating with digital information in a co-located setting. In this scenario, the user on the far left has found an interesting piece of information that she wishes to share with the group. To share the information, she turns her laptop screen to be viewable to the group. The user on the right then points in an effort to get the laptop's owner to select an item on the screen to retrieve more detail on a topic of interest. Because the user on the right cannot read the information on the screen, she also has to ask one of her peers to read the content. When an item is found that is interesting to the user in the center, he asks the laptop's owner to email a URL to the resource.

Unfortunately, many of the tools and supporting devices that are currently used to facilitate the use of digital information are designed for use in a single user, single display environment (exemplified in Figure 1.1). For instance, it is still very difficult with current tools for a group of users to jointly edit a digital document at the same time. The high

overhead required for users to exchange, share, create, and organize digital information artifacts while engaged in collaborative work often prohibits their use.

In contrast to digital artifacts, use of physical information artifacts like whiteboards, printouts, and notebooks are effective in supporting collaborative activities. Consider a group which comes together for a brainstorming session where whiteboards and loose sheets of paper are used. In this environment, any user can go up to the whiteboard and provide shared information to the group, users can exchange ideas by passing around sketches on the sheets of paper, and users can spread out these sketches on a tabletop to compare and organize their ideas. This example illustrates how easily the use of physical artifacts integrates into the collaboration practice of the group. The contrast between physical and digital artifacts highlights a fundamental tension between the value of digital artifacts in collaboration, and the simplicity and naturalness of collaborating with physical artifacts.

To reduce this tension, better solutions are needed to enable users to more effectively utilize and share digital information artifacts when engaged in co-located collaboration. In this dissertation, we investigate multiple device environments, or MDEs, as one promising solution path for supporting collaboration with digital artifacts. As illustrated in Figure 1.2, the concept of an MDE comprises a physical workspace (e.g. conference room) that contains co-located personal (e.g. laptops) and shared devices (e.g. large displays) that are networked to form an integrated workspace [133].

MDEs offer many potential benefits for improving how groups collaborate. Such benefits include the ability to place myriad information artifacts on shared displays for comparing, discussing, and reflecting on ideas; to jointly create and modify information to enhance focused problem solving or enable serendipitous collaboration; and to allow quick and seamless transitions between individual and group work.

Figure 1.2: A typical multiple display environment, or MDE. Notice the presence of the large, shared displays. In an MDE, these are useful for sharing and jointly interacting with information artifacts as a group. Personal devices are also present in the workspace. These laptops, tablets, and desktop PCs can be used to maintain access to private, personal information and workspaces. With appropriate infrastructure and user interfaces, an MDE has the potential to greatly improve how users include the use of digital information artifacts in their day-to-day collaborations.

A key advantage of MDEs is their potential to provide users access to both private and shared workspaces during collaborative tasks. This quality is an essential requirement for effective co-located collaboration because it allows users the ability to perform independent, individual activities while participating in group activities [44, 90, 124]. Such situations are common; for example, a user could be keeping personal notes, researching a topic independently to report back to the group, or simply just maintaining awareness of ongoing activities that are not related to the current collaboration.

A crucial challenge is developing software frameworks that provide the necessary user interface and system level support to enable users to fully realize the potential benefits of MDEs. For example, support for facilitating the replication of information content and enabling joint interaction by all group members.

The need for such frameworks is rapidly increasing. Workspaces equipped with a shared large display are pervasive in today's work places. For example, nearly all modern conference rooms are equipped with at least one LCD projector or large display. Combined with the affordability of personal devices such as laptops and tablet PCs, organizations are now easily achieving the physical and hardware requirements of MDEs, but still lack adequate software solutions for leveraging their full value.

This dissertation contributes the design, implementation, and evaluation of a new interaction framework, called IMPROMPTU, which enables users to better realize core principles of effective group work when working in MDEs. Critical to the success of this work is the research process taken. This included leveraging and integrating knowledge from fundamental theories of collaboration, understanding existing collaborative practices related to the use of MDEs, and applying principled HCI design techniques.

It is through this approach that we were able to create a novel solution that provides important capabilities not seen in existing MDE frameworks. For example, IMPROMPTU provides a new user interface that provides improved activity awareness within groups and facilitates serendipitous collaboration. Its application sharing services are the first to enforce negotiated sharing strategies and controls, providing users a natural and intuitive process for establishing and managing shared applications. IMPROMPTU's application sharing is also extremely flexible, allowing simultaneous interaction with shared applications without affecting the local input actions of other users. The value and impact of these contributions are proven in a field study investigating IMPROMPTU's use in authentic task domains.

1.1 Existing Approaches

Many research projects have developed systems and/or frameworks for MDEs (e.g. [73, 108, 133, 134, 136]), but all fall short in allowing groups to realize fundamental principles of effective group work. For example, with Colab [Stefik, Bobrow et al. 1987], groups can only work with digital information supported by custom built applications, significantly limiting the types of information that can be shared. In the iRoom [Johanson, Fox et al. 2002], relocating applications (information) does not maintain interaction context (e.g. stack traces in a debug window would be lost). This impacts users' ability to efficiently transfer information between devices, hindering their ability to seamlessly transition between individual and group activities.

More broadly, all of these existing frameworks were designed from a system-centric perspective; where researchers concentrated more on satisfying system-level challenges and less on the challenges of supporting collaboration within authentic task domains. We took a different approach with the design of IMPROMPTU. Building on fundamental theories of collaboration and an understanding of collaborative practices related to the use of MDEs, we develop a framework that not only affords effective sharing of everyday information artifacts (i.e. software applications) between devices, but allow this sharing to occur in a manner consistent with users' expectations and existing collaborative practices.

Aside from MDEs, there are other physical configurations of technology and supporting groupware solutions aimed at co-located collaboration. One alternative configuration is Single display groupware (SDG). SDGs are specific applications or systems that allow multiple users to simultaneously interact with digital information loaded on a single, shared display. SDG systems like Dynamo [68], the SDG Toolkit [144], and others [95, 134] have shown to be effective in providing users a seamless shared electronic workspace where each user has equal access to digital information. With SDG systems, it becomes possible for users to load existing digital artifacts for group review and edit, collaboratively explore knowledge sources such as websites or databases, draw within a shared canvas, and more.

Unfortunately, SDG systems do not allow users to maintain an individual workspace that can remain private (not viewable by fellow collaborators) when desired. This reduces the types of collaborations that can be supported by SDGs. Additionally, even for the tasks that are supported, the single shared workspace configuration allows known inhibitors of effective collaboration to exist [90]. For example, evaluation apprehension can prevail because each user's work is created in the open; enabling other members of the group to judge an individual's work before it is complete [36].

Tabletop systems extend the advantages of SDGs to a more natural configuration – a horizontal work surface. Exploiting users' familiarity with working with physical artifacts on tabletops and desktops, tabletop groupware systems provide many key advantages, which include the ability for users to easily transition between roles, allows users to maintain awareness of other user's actions, encourages exploration, and more [114]. Tabletop systems like UbiTable [127], DiamondSpin [128], MultiSpace [47], and others demonstrate the ability of tabletop systems to support co-located collaborative tasks. However, like SDGs, tabletops also do not provide users private workspaces. Further, tabletops also suffer from a limited effective work area, do not allow shared orientation of artifacts without requiring physical movement, and do not scale well as group size increases.

Absent of any specific groupware systems, collaboration with digital information often falls into two situations: 1) a user interacts with her personal device, such as a laptop or tablet, to access information and then verbally shares that information with the group, turns her screen to share information with the group (as demonstrated in Figure 1.1), or emails links or attachments to other collaborators; or 2) one user connects her device to a shared display, like a LCD projector or plasma display, and shares her local desktop and applications with group. In both situations, users are severely restricted in their ability to share digital information, significantly reducing the value of utilizing digital information in support of collaborative activities.

1.2 Our Solution

In this dissertation, we present a new, fully-functional interaction framework called IMPROMPTU (Improving MDE's Potential to support Tasks that are genUine). A key technical innovation of the framework is to enable users to replicate any off-the-shelf application between devices, to jointly interact with replicated applications, and to quickly place and interact with applications on shared large displays. Additionally, the framework provides an easy, intuitive interface that was specifically designed to facilitate quick and low overhead interaction with underlying MDE functionality.

The functionality provided by IMPROMPTU allows users to leverage the utility of MDEs in ways that were never before possible. The flexible application sharing in IMPROMPTU is the first groupware system for MDEs that fully supports users' ability to quickly access both private and shared workspaces. Also crucial is that users are able to easily manage the information across these workspaces, significantly reducing the cost of using digital information in collaborative activities. Additionally, IMPROMPTU is unique in its ability to support a multitude of collaboration modalities, seamlessly transition between those modalities, and allow users to work in multiple modalities at once. With this flexibility, IMPROMPTU does not force users to work in predefined ways, but rather can be adapted as the collaboration naturally progresses.

In designing IMPROMPTU, we followed a user centered design process that included surveys and interviews with current and potential MDE users. Through this process, we gained an understanding of the common collaborative activities users perform (or desired to perform) within MDEs, what information is used and exchanged, and how existing technologies are leveraged. This understanding was pivotal in grounding the design of IMPROMPTU within the context of how users currently collaborate. Further, this design process alone produced lessons that further reinforced the limitations of existing solutions.

We have made the framework's source and executables publicly available for download, allowing others in the community to use it in their own collaborative and research

activities. We designed the interaction components of the framework to be easily decoupled, permitting the core replication and input services to be used absent the interface. This affords the opportunity for other researchers to leverage and/or extend these services to support a broad range of collaborative work (from co-located to distributed). They can also develop and test new interfaces and interaction techniques on top of the core services.

The value of IMPROMPTU to support every day, real world tasks was demonstrated in one of the first field studies of MDE use within an authentic task domain. The framework was deployed within two software development teams over a three week period to support their daily work activities. Results showed the framework supported users performing a wide range of problem solving activities. Most importantly, the framework was able to effectively support the information sharing needs of users while also allowing them to perform collaborations using their own customs and practices.

1.3 Illustrative Scenario

We provide an illustrative scenario that further demonstrates how the contributions of this dissertation improve co-located users' ability to collaborate effectively with digital information. In this scenario, a team of five software developers are co-located within a multiple display environment. Each developer has his own desk and personal workstation; there are also two large shared displays present in the workspace. The team is just about to meet and plan the next set of functionality for their project.

Before the meeting begins, each developer uses Microsoft Word to prepare his own list of items that he feels needs to be discussed at the start of the meeting. At the designated meeting time, the lead developer utilizes IMPROMPTU to replicate his list onto one of the large displays and discusses his items with the rest of the group. As they go around the room, other developers share their lists on the second shared display. In this way, the team can compare the items on each list side by side as a group. As relevant items on each developers list are discussed, input redirection to the large displays is used to allow information on one list to be copied and pasted to lead developer's master list.

For certain discussion items, the developers may also want to demonstrate a concept (i.e. a bug or compiler error in Visual Studio, a Visio diagram illustrating the design of a new data structure, or a Photoshop mockup of a new UI component). Thus, prior to the meeting the developer places the relevant application windows on the large display so the information can be easily shared with the group.

For example, to discuss possible resolutions to a complex bug, a developer replicates the code on one shared display and a running instance of the code on the other shared display. As the bug is demonstrated, other developers create a local replication of the other user's code editor on their personal device. The developer can then modify the shared code while the rest of the team follows his modifications in real-time using the replication on the shared display.

As the meeting draws to an end, the developers end replication of shared information on the large shared displays. They however leave the master list shared on one of the large displays. As they work individually to complete items on the list, each developer has access to replicate the list onto his local machine to update. In this way, the list is always current and provides the group with an ongoing awareness of current project status.

The scenario above demonstrates many of the features provided by our framework. IMPROMPTU allowed users to easily transition information from their private workspaces (i.e. personal devices) into the group's shared workspace (i.e. large displays). This allowed users to easily transition their digital information artifacts between individual and collaborative tasks in ways akin to how physical artifacts are shared during collaborative activities. Users also seamlessly shared interaction capabilities, allowing shared information to be quickly and easily changed in support of the collaboration. Additionally, IMPROMPTU allowed the team to maintain ongoing awareness and access to shared information. Far exceeding the capabilities of existing solutions, our framework provides advanced sharing capabilities that are easy to use and, most importantly, support most common collaboration modalities.

1.4 Scope

Our work focuses on supporting small groups of approximately 2-8 people working at the same time in the same physical workspace. We also make the assumption that these workspaces have a relatively small number of shared displays (1-3) and that each user has access to his or her own personal device (e.g. laptop or tablet). These configuration characteristics represent most of the commonly used configurations used in practice today [48, 90].

There are of course other configurations in which people can collaborate. For example, users can collaborate at same time, but in different places; or even at different times and different places [71]. We focus on supporting co-located collaboration as it is the most common forms of collaboration that occurs in today's workplace [91]. Additionally, designing for these other configurations require different and/or additional design assumptions to be made and likely require different end solutions. However, as we discuss at length in later chapters, many of the lessons and design principles produced through this work likely have some applicability to other configurations.

There are also many non-technical factors that influence collaborative work. As Axelrod explains, successful cooperation largely depends on equal reciprocity between all participating members [12]. That is, to benefit the most, all parties must receive some gain from participating in the collaborative activity. In fact, the most critical components of collaboration are by and large a factor of group dynamics [43, 48]. Group members must be diverse enough to contribute ideas from different perspectives [69], need to be respectful and open to contrasting opinion [36], creative and reinterpretive with others' ideas [66, 86, 122], and much more. In this dissertation, we do not provide any explicit support for promoting effective group dynamics. We assume that the groups we are supporting are individuals who are comfortable and sufficiently motivated to work together.

1.5 Summary of Contributions

This dissertation makes the following contributions:

- *Results from a contextual inquiry and iterative prototyping investigating the use of MDEs to support authentic collaborative activities*. Much of the existing work on MDE interfaces and systems has been largely focused on supporting short-lived, artificial tasks. In this dissertation, we sought to support the common, everyday collaborations that are performed within authentic task domains. To determine expectations of use in these collaborations, we performed an in depth contextual inquiry that consisted of surveys, interviews and iterative prototyping with existing and potential users of MDEs. Through this process we derived a core set of system requirements for supporting users' collaborative needs. These requirements can be used to guide the design and development of new frameworks to support collaboration in MDEs. We discuss in detail the process we followed and the result requirements in Chapter 4.

- *A fully functional interaction framework for supporting collaboration in MDEs*. Building upon lessons gained from understanding users' information sharing needs, IMPROMPTU is one of the first groupware systems to enable users to effectively utilize the most important collaborative capabilities of an MDE [24]. A major innovation of IMPROMPTU is its ability to provide access to private individual workspaces and shared workspaces at the same time, and to fluidly transition information between these workspaces. Other important innovations include a people-centric user interface that provides activity awareness and facilitates serendipitous collaboration, sharing services that enforce negotiated sharing strategies and controls, and functionality that allows simultaneous interaction with shared applications without affecting local input actions. The framework has been made publicly available, allowing others to form and use MDEs to improve their own collaborations. A detailed description of the framework's design and functionality is provided in Chapter 5.

- *Results and lessons demonstrating the effectiveness of the framework within an authentic task domain*. Prior studies have shown the utility of MDEs to support collaboration within controlled and often contrived settings. We performed one of the first field studies that investigated how a fully functional MDE framework supports collaboration within an authentic task domain [24]. The findings of the study provide an evidence-based understanding of how an MDE is leveraged to support collaboration, and highlight some of the complexities in designing frameworks to support users in practical settings. We provide new design lessons for MDE frameworks and provide directions for further research in this space. We present these details in Chapter 6.

- *A new interface metaphor for managing information artifacts in MDEs and empirical results demonstrating its efficacy relative to existing solutions.* Many interfaces exist for managing information artifacts in MDEs. However, many of these interfaces do not enable users to perform all of today's necessary management tasks, do not support common input mechanisms present in an MDE, or use difficult to understand representations of the workspace. To create a more effective interface, we developed the world-in-miniature interface (or WIM) metaphor for managing information artifacts within an MDE [19, 22, 23]. Our WIM interface enables a user to *visually* relocate artifacts among screens, whether those screens are portable or fixed, small or large, or near or far from the user. Evaluations comparing the WIM interface to current state-of-the-art interfaces show that the WIM provides more efficient interaction, reduced error, lower subjective workload, and higher user satisfaction [20, 21]. From this work, we also produced one of the first sets of principles for designing interfaces for MDE [21]. These resulting principles were integrated into the design of the interface for IMPROMPTU. We present this work in Chapter 3.

Chapter 2

Related Work

In this chapter, we discuss existing research as it relates to the contributions of this dissertation. We begin by reviewing relevant theoretical research related to collaborative process and practice – detailing the evidence this past work has amassed demonstrating the utility of collaborative work and what is necessary for it to be effective. We also review core functionality requirements for effective groupware systems and tools.

We then situate our work within the large body of existing co-located collaborative systems; detailing past lessons that we build off of and discussing limitations. We then survey existing work within the specific area of multiple display environments. We detail the alternative approaches that have been taken, explain their limitations, and compare them against solution paths taken in this dissertation.

We also discuss evaluations of co-located groupware systems and how our evaluations build from and/or relate to these existing studies. We end by providing background for the target task domain that we concentrate on in this dissertation: group-based software development.

2.1 Co-Located Collaboration Benefits and Inhibitors

Researchers in psychology, social psychology, communications, and organizational management have spent decades investigating collaborative practices. This research has repeatedly shown the value of co-located collaboration in the workplace. Brown and

Palinscsar show that individuals who work together are more effective at eliciting problem definitions, decomposing problems into practical units of work, and monitoring the work to determine if a workable solution has been found [29].

These benefits are likely due to the fact that individuals frequently build off of each others' ideas and opinions during collaborative work. Often referred to as reflection [122] and reinterpretation [66, 86], these processes have been shown to induce a high level of cognitive thought [51]. As a result, working in groups often produces more, higher quality ideas compared to what is produced when working individually [41, 101].

Additionally, there is evidence that working in groups enhances learning [142], increases appreciation of different perspectives [10, 78], and even improves trust and sense of belonging to the group [50, 143]. Because of all of these benefits, increasing collaboration is seen as a strategic initiative in many organizations [10, 13, 102, 103, 105]. For example, many software development organizations are striving to lower barriers for collaboration as a means for improving the quality of software created and the efficiency of the development process [83, 123].

Despite these benefits, there are some significant inhibitors that researchers have found to impact the overall effectiveness of collaborative work. Many individuals participating in group activities choose not to contribute ideas to the group for fear of being negatively evaluated by the group [36]. Others experience social loafing or the tendency to reduce their contribution to the group effort as the group size increases [81]. This occurs more often with simple tasks as participants do not feel it takes a "full group effort" to complete those tasks [77]. In efforts to reach consensus, groups sometimes tend to follow a specific solution path early. Referred to as groupthink, its consequence is that favorable, and even optimal, solutions may never get discussed in the group [69].

All of this past work is important and relevant to this dissertation. First, the research in the social sciences has shown the productivity, social, and learning value that collaboration creates during problem solving activities. There is therefore an incentive to adopt frameworks like IMPROMPTU to enable and encourage face-to-face collaboration.

Second, it also identified important inhibitors known to reduce the effectiveness collaborative work. Thus, this past work identifies the *minimal* set of design requirements that collaborative systems and tools must provide to be effective.

2.2 Utility of Artifacts in Collaboration

Collaboration often involves the use of information artifacts such as documents, spreadsheets, examples, task lists, etc. Several research projects have looked more closely at the use of information artifacts and their role in collaborative activities. From observations of co-located collaborative problem solving, Tang found gestures were an important communication mechanism between collaborators, the process of creating artifacts was often more important than the resulting artifacts, information artifacts were essential to motivation and mediation of the group, and that spatial relationships among users would often determine user's role within the task [140].

Other studies found similar results. Brinck and Gomez found that notes and quick drawings on whiteboards provide rich semantic meaning and aid users in maintaining and recalling the history of their collaboration [28]. Other studies have also shown the importance of mutual knowledge and coordination in groupwork [79, 125].

Olson et. al. studied programmers performing early stage software design meetings [99]. Their study found that users would frequently switch context, often requiring information artifacts to be rearranged or reconfigured. Their study found that managing information artifacts consumed as much as 20% of the total time spent on the collaborative task. This is a significant amount of overhead that takes away from the time groups spend to create, review, and debate ideas and solutions.

Researchers have found users not only need more effective ways to manage information artifacts, but these mechanisms also need to address the social aspects of shared information access. For example, Morris et. al. found that standard social protocols are insufficient in preventing coordination and access conflicts in co-located groupware [94]. Scott, et. al. discuss similar results for their investigation of tabletop systems [124]. The non-conformity to social standards is likely a result of several factors, which include: use

of digital information tools is primarily an individual activity, causing individual interaction behaviors to be inadvertently transferred into multi-user tools; and, the novelty of digital collaboration tools often leads to users performing actions in error which inadvertently conflict with other users.

This past research has shown the exceptional value and importance that information artifacts have in collaboration. Availability and accessibility of information artifacts are thus paramount considerations in the design of systems and tools to support collaboration. Further, more recent work has shown the importance and complexity of the design space for creating collaboration tools that uphold social conventions when working with shared information.

This dissertation applies insights and lessons of this past work to build a framework that reduces the overhead for users to create and share digital information artifacts within multiple display environments. Considering the near ubiquitous use of digital information tools used in today's workplace, our framework can greatly improve a wide variety of collaborative activities.

2.3 Requirements for Effective Co-Located Groupware

With the many advantages co-located collaboration has to offer, researchers have sought to better understand how to build software systems that allow users to effectively utilize digital information artifacts in their activities. Informed by decades of research designing, implementing, and evaluating groupware, several lists of requirements have been developed (e.g. [44, 90, 124]). In this section, we synthesize nearly all of the requirements provided by this past work into a single, master list of requirements. The only exceptions to inclusion in this list are requirements that only applied to a narrow groupware configuration and not groupware in general.

- *Support for the user to transition between personal and collaborative tasks*. As discussed in [44], there are two types of transitions that must be supported. First, users need to be able to shift between private work in their personal workspaces and collective work in shared workspaces. Second, users need support to

Requirement	SDG	Tabletop	MDE
Easily Transition Between Individual and Group Work	○	◒	★
Access to Private, Individual Workspaces	○	○	●
What You See is What I See (WYSIWIS) with Shared Perspective	●	◒	★
Low Interaction Overhead	●	●	★
Adherence and Enforcement of Social Protocols/Structure	◒	◒	★
Support Multiple, Simultaneously Occurring Tasks and Fluid Transitions When Switching Tasks	◒	◒	●
Support Multiple Collaboration Methods	○	◒	★
Support Flexible and Natural Arrangements of Users	◒	◒	●

Legend: ● = Full Support, ◒ = Limited Support; ○ = Little to No Support

★ = Full Support with Contributions Provided in this Dissertation

Table 2.1: A summary of important requirements for effective co-located groupware. The table also summarizes how existing groupware solutions support these requirements.

transition between private work and collective work while engaged in group collaboration.

- *Access to Private, Individual Workspaces*. Access to private workspaces with personal information artifacts is necessary to promote users' ability to independently explore information or perform subtasks during group collaboration [44, 90]. Similar to having access to a personal notepad, similar support needs to be provided when working with digital information artifacts. Further, working in a private workspace reduces feelings of evaluation apprehension (which would occur if each user's work was always viewable) [36].

- *Low Interaction Overhead*. Groupware systems should be designed so that users do not have to spend a large amount of time and effort to configure and maintain the workspace. Thus, interacting with the groupware should induce low mental workload and provide intuitive, efficient interactions [124].

- *Adherence and Enforcement of Social Protocols and Structure*. As discussed in the previous section, social and cultural protocols and structures largely impact the success of a collaborative activity and, as a result, effective protocols and structures should be directly supported by groupware systems [90]. For example, systems should enforce social concepts of artifact ownership and use.

- *Support Multiple, Simultaneously Occurring Tasks and Fluid Transitions When Switching Tasks*. Collaborative activities are often composed of smaller activities that are performed in parallel. For example, in a brainstorming design session a group of designers might work on several designs at the same time, often taking an idea from one to add to another. To support such activities, groupware systems need to allow for multiple activities to occur in tandem, and the ability to access information artifacts across all tasks at the same time [44, 90].

- *Support Multiple Collaboration Practices*. To accomplish a group task, several collaborative practices may be employed. For example, a set of engineers may brainstorm early in a collaboration session, transition to reviewing, and eventually collaborative writing. Thus, to support the entire collaborative session, a groupware system must support all these practices and the ability to easily transition between them [90].

- *Support Flexible and Natural Arrangements of Users*. Users often arrange themselves physically within a workspace in ways that are the most natural and comfortable to the users. For example, users when sitting at a table, users often sit across from one another so that they can easily look at their collaborators when they speak. Thus, groupware systems should not constrain users' physical

Figure 2.1: A screen shot of the Dynamo System. With Dynamo, multiple people can jointly interact within a shared work surface using their own input devices (see upper left). While Dynamo allows users to *lock* content, it does not provide a mechanism for a user to work on tasks in a private, non-group-viewable workspace.

configuration so much that collaboration is no longer natural or comfortable for users to engage in [124].

2.4 Groupware Support for Co-Located Collaboration

Researchers in the fields of HCI and CSCW have made significant strides in the development of systems and frameworks to support effective co-located collaboration. Existing solutions comprise three main physical configurations of technology: single display groupware, tabletop systems, and multiple display environments. Table 2.1 provides an overview of how each configuration, and its supporting groupware, satisfies the requirements discussed in the previous section.

2.4.1 Multiple Users on a Single Display

Single display groupware (SDG) allows co-present users to concurrently interact with applications on a single screen. Several research projects have investigated the use of SDG to support collaboration. For example, KidPad allows multiple children to draw on the same canvas [135]. Pebbles allows multiple users to interact with an application on a shared display through PDAs [95].

As demonstrated in Figure 2.1, Dynamo allows users with their own keyboard and mouse to work collaboratively on a public interactive surface comprised of one or more large displays [68]. In their tool, users can bring in their own information artifacts on personal media (e.g. USB memory key) and add it to the shared workspace. Ownership of artifacts is also supported in Dynamo through functionality that allows users to place identifying photos or icons next to their applications. Tse and Greenberg have created the SDG Toolkit which allows developers to create applications that can support simultaneous interaction from multiple users [144].

While effective for supporting some collaborative activities (e.g. those that are highly coupled and synchronous), SDG fall short of supporting all of the core requirements for effective groupware (discussed in the previous section). For example, because all users must work on the same work surface, the users are not able to work on individual tasks separate of the collaborative activity. This limitation extends to inhibit certain collaborative modes of work. For example, the ability to work privately to reinterpret or revise another user's idea (a core activity of creative brainstorming [66, 86, 122]) is not supported. Further, because users' actions can always be viewed by their peers, some users may experience evaluation apprehension, the fear of others judging one's work before it is intended to be shared [53]. Because of these numerous limitations, we chose to explore MDEs, rather than SDGs, because MDEs have a better prospect for supporting the full set of groupware requirements.

Figure 2.2: The UbiTable from MERL allows users to jointly interact around a shared horizontal work surface. The green and pink areas represent private work areas – a workspace where only the user on that side of the table can interact with the content.

2.4.2 Tabletop Surfaces

Tabletop displays allow users to interact with digital information on a horizontal surface. As shown by Rogers and Lim [113], horizontal work surfaces allow users to easily transition between roles, supports strong awareness of other user's actions, encourages users to explore, and more. Many systems have been developed to enable and enhance these advantages.

As shown in Figure 2.2, the DiamondSpin toolkit [128] and its supporting applications like UbiTable [127] allow users to perform collaborative tasks easily on a multi-user tabletop surface. Extension of this work allows users to migrate digital information artifacts to other vertical devices in the workspace [47]. Other systems like ConnecTables

[139] allow users to physically join two horizontal displays to form a temporary unified workspace where users can drag information between the displays to exchange them.

Tabletops support many of the requirements for effective groupware (summarized in Table 2.1). However, like SDG, they fail to provide users with access to their own private, individual workspace. Additionally, for many tasks, a horizontal work surface is not always conducive; e.g. users would need to crowd around one end of the table to share the same perspective. While there has been some work to mitigate these issues, they still persist. Because of these fundamental limitations, our work focuses on MDEs, as they offer the most potential for supporting the widest range of collaborative activities and workspace configurations.

2.4.3 Multiple Display Environments

Multiple Display Environments (MDEs) are physical environments that are comprised of personal devices (e.g. laptops and tablets), shared large displays (e.g. plasma and LCD panels), and specialized software infrastructures which connect the independent devices to provide a unified shared information space. An MDE affords users the ability to have shared information space via the shared large displays, while also maintaining access to private work areas via their personal devices. Because they can be combined from existing independent devices, MDEs could even include shared horizontal displays (as used in the tabletop systems above) and allow users to leverage the benefits of horizontal display configurations (e.g. see discussion of benefits in previous section).

To illustrate the benefits of an MDE, consider the following motivating scenario where users are working together to plan an upcoming trip. In the scenario, one user researches independently on his personal laptop air travel options while another investigates hotel information on her laptop. When one user finds an item of interest, he or she relocates that information to one of the large displays so the information can be shared and discussed as a group. Alternatively, the group could join together and start searching for information together using one of the large displays. When multiple options are found, the users can organize their options across the devices. Perhaps one shared display being the option with the most frequent flyer points and the other being the least expensive.

While relatively simple, this example illustrates some of the key advantages of MDEs. In particular, an MDE can provide two important advantages not supported by other groupware configurations: 1) the ability for users to have both shared and personal workspaces within the collaborative environment, and the appropriate interaction techniques that enable effective transition of artifacts between them; and, 2) the ability to employ a wide variety of collaboration modalities to group, organize and collaboratively interact with shared information artifacts spread across the multiple display surfaces.

MDE also fulfill many important aspects of Weiser's *Ubiquitous Computing* vision [145]. For example, Weiser and his colleagues at Xerox PARC prototyped a system that was composed of what they called *tabs* (small PDA sized devices), *pads* (tablet or laptop sized devices) and *boards* (large vertical shared displays). With the ability to share information seamlessly across these devices, Weiser and colleagues were essentially prototyping an early MDE framework. However, a fundamental deference between Weiser's tabs, pads, and boards, and the modern definition of an MDE is the fact that Weiser assumed device use was ephemeral. That is, devices would be acquired as needed and discarded after use. MDEs, on the other hand, assume a high degree of ownership and continued use of personal devices – even so much as they are specifically designed for personal devices, and the information contained on them, to be easily transitioned for use between individual and collaborative activities.

This dissertation seeks to make the affordances of an MDE better realized within a functional set of software frameworks and user interfaces. In the later sections we further discuss existing work in MDEs. We divide our discussion into two main categories: infrastructures and frameworks for MDEs, and interfaces and interaction techniques for MDEs. In each section we explain how existing solutions have sought to support core groupware requirements, their limitations, and how work in this dissertation differs, as well as contributes to advancing the current state-of-the-art.

2.5 MDE Infrastructures and Frameworks

Distributed infrastructures such as Gaia [116], iROS [72] and Aura [131] provide systems-level services for application relocation and file sharing in an MDE. Gaia, for example, supports presence detection for users, devices, and services, provides context events and services, and supports an information repository for entities in the workspace [116]. However, to take advantage of many of the features in these infrastructures, specialized applications need to be built. For example, to relocate an application window in Gaia, the application must be built to run on Gaia's application framework [115, 117]. This requirement is a major hindrance since a user cannot use the familiar applications used to support their individual tasks to support their collaborative tasks.

Tools like WinCuts [138] allows users to replicate a local window's pixel data to other devices (e.g. large shared displays). This allows sharing of task information, but does not allow for joint interaction by two or more users. LiveMeeting [5][5][5] and Community Bar [141] allow users to share each other's desktop screens and interact with them. However, these systems do not support the ability to share applications from multiple devices at the same time. For example, this would not allow two users to view and compare each other's ideas in parallel.

Other tools like SAME [70] and SharedX [52] provide the ability to share X Windows Applications. While these systems do provide the ability the share any application built for the X Windows operating system, they lack usable interfaces that provide the ability for users to control how and when their applications are available to peers.

In contrast, to the above tools and systems, the Colab [133] and CoWord [151] projects seek to provide multi-device support for collaboration on a per-application level. For example, in Colab, there is a specific application for facilitating group note taking. Like the above described MDE frameworks, a major limitation of this approach is that user cannot incorporate existing applications *and* information into a collaborative activity. This is a significant hindrance as it does not enable users to easily transition between individual and collaborative activity. Further, a per-application approach would also

require MDE developers to create a new application suite for each activity being supported. This is a substantial amount of overhead and would impede the flexible characteristics of an MDE to adapt to different activities and collaboration styles.

As shown in Table 2.1, this dissertation makes many contributions to realizing the benefits of MDEs. We present a new interaction framework that was built from the ground up that overcomes the limitations of previous frameworks. For example, our framework enables users to share any off-the-shelf application, enabling users to leverage existing tools for both individual *and* collaborative tasks. This also allows information to be more easily transitioned between private and shared workspaces. Additionally, the framework provides flexible models of application sharing, enabling any number of applications to be simultaneously replicated across any number of devices. This enables users to share applications from multiple users at the same time (perhaps on a shared display), perform true joint interaction, and allows them to use and arrange information in ways that best suits the group's collaboration style.

2.6 MDE Interfaces and Interaction Techniques

Many user interfaces and interaction techniques have been developed for managing applications and input among independent devices. We organize our discussion around the basic interaction themes that this past work comprises.

2.6.1 Traditional Graphical User Interface Techniques

Traditional graphical user interfaces that use textual descriptors to represent users, devices, and applications are the most common interfaces for managing MDEs. iCrafter [104] and Gaia [116] enable a user to relocate applications between devices through selection boxes to specify the application's name, source device, and destination device. In [75], researchers extended a Web browser to enable users to relocate browser windows across machines. Loading a website on a particular device is performed by the user selecting a textual identifier of the destination screen from a list of available choices. In Colab [133], users are provided an interface to specify which shared application is displayed on the large display.

While sufficient for enabling users to manage applications and devices within an MDE, their use is inefficient and error prone [20]. This is because these interfaces require users to constantly map devices and applications to textual descriptors. As we show in this work, interfaces that represent devices and applications in a visual representation enable faster and more accurate interaction.

2.6.2 Physical and Vessel Based Techniques

Several research projects have investigated the use of physical movement of people and devices to organize and control where information is displayed. In Easy Living [30], the managing infrastructure actively tracks users in the workspace and automatically relocates applications to devices closest to the user. With Pick-and-Drop [106] (shown in Figure 2.3) a user can pick up an object on one display with a stylus, then drop it on a different display to relocate that object.

With ConnecTables [139] and the Stitching interface [67] users can place two devices next to each other to form a shared continuous display workspace. Users can then easily drag and drop artifacts between devices using this shared workspace. In I-Land [136] users can shuffle, throw, take, and pick-and-drop applications within large displays and between personal devices.

The use of physical objects and movement as an interaction technique lowers the overall interaction cost since users do not have to map interface representations of people and objects to their physical instantiations. However, requiring physical movement can be awkward in many collaborative situations. For example, exchanging a document between two users sitting on opposite ends of a conference table will require at least one of those users to unnecessarily move about the workspace. In contrast, the interfaces in our work allow for quick mappings between the interface and physical representations and do not require physical movement to operate.

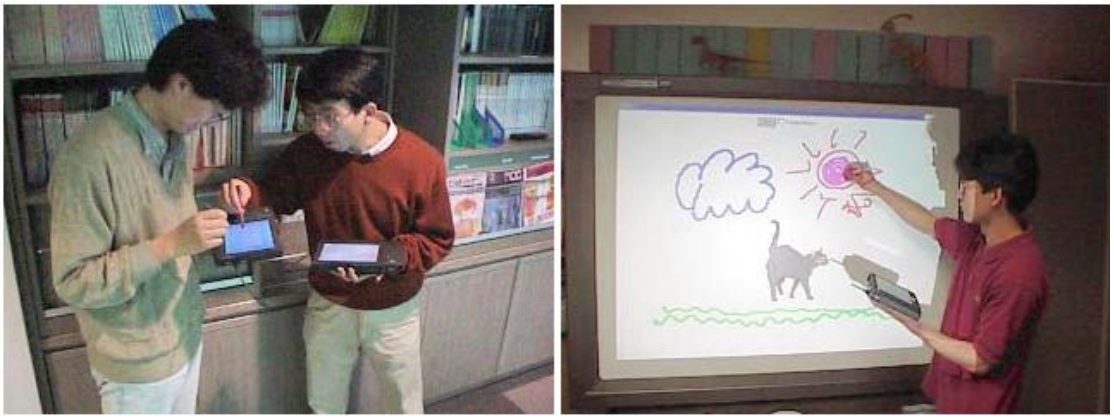


Figure 2.3: The Pick-and-Drop interface allows users to select content on one device (pick) using a stylus and then place that information on another device by touching the stylus on the destination device screen (drop). Notice how the interaction requires the user to be within reaching distance to both source and destination devices (e.g. to relocate to a large screen the user has to physically stand in front of the large display to complete the interaction).

2.6.3 Virtual Path Techniques

With UbiTable [127] and augmented work surfaces [108], users are able to share applications among personal devices as well as shared displays. The main interaction technique for sharing applications is the use of a virtual path, where users select and drag an application to the edge of a display causing it to appear on the display of an adjacent device.

PointRight uses geometric paths to enable input redirection across devices [74]. Users can move the cursor directly (without a UI control) between local and shared devices, and interact with applications. MightyMouse offers a similar concept of redirection, but

provides a control panel with buttons positioned relative to the spatial layout of the corresponding devices in the workspace [27].

The E-conic interface enhances the virtual path interaction by providing perspective-aware paths [97]. With E-conic, the virtual paths are unique to each user and are based upon their location within the physical workspace to make the connections more logical for each user.

While effective, virtual path interfaces are not compatible with many input mechanisms commonly used in multiple display environments. Stylus and touch input do not allow users to traverse a virtual path since input can only be performed directly on display surfaces. Thus, building an MDE based on virtual path interaction techniques significantly limits the types of devices that can be used effectively.

Existing virtual path implementations also do not provide the user an awareness of how devices are connected. Thus, users need to discover and memorize paths between devices. This becomes significantly difficult when the paths are not overly intuitive, a large number of devices are present in the workspace, or devices are consistently moving or being rearranged.

2.6.4 Map Interfaces

While map interfaces have been used to manage application windows on personal devices (e.g. see [3, 4, 7]), they are also an emerging as an effective interaction technique for managing content in MDEs. With a map interface, devices, applications, and even users are shown in a spatial layout that represents the physical layout of the workspace . For example, in ICrafter [104] users are provided a strict, top-down view of the devices within the workspace. Users can select a device to get access to different services, settings, and content on those devices. In this dissertation, we propose the world-in-miniature interface metaphor which provides users a visual representation of the devices, applications, and other salient features of the workspace such as walls, tables, and doors. Evaluations of our interfaces show the spatial representation provides improved workspace awareness, better performance, and increased accuracy [21].

2.7 Techniques for Window Management

Building techniques to manage application windows on a single user, single display system has been a heavily researched topic within the community. Important to this dissertation are the lessons from this existing work which can be extended and/or adapted to managing windows across multiple displays.

One of the first advanced window management tools was the Rooms virtual desktop manager [64]. With Rooms, a user could create a virtual desktop for each of his open tasks and organize task related application windows across those desktops. For example, a virtual desktop could be created for email with one's Outlook window along with Hotmail loaded in a web browser window. Another virtual desktop for writing could be created with open windows of Word, EndNote and Acrobat. Users can then switch between rooms as they work on different tasks. The Rooms system allowed users to better organize windows by organizing them into smaller, more manageable groups. However, Rooms has one significant limitation; users are forced to work with a task's windows in mutual exclusion of another task's windows. This interaction model prevents users from maintaining an awareness of the information in other tasks, and from being able to work on multiple tasks at once. Subsequent projects have brought the virtual desktop metaphor to modern operating system implementations[1, 3, 4, 7][1, 3, 4, 7][1, 3, 4, 7]; but still lack awareness and inter-task interaction mechanisms.

More recently, researchers have explored managing windows using 3D environments [109, 110], zoomable workspaces [16] and time-based representations [107]. Each of these approaches use visual effects to provide users more work area to spread out and organize windows. With Task Gallery [110], users can organize application windows within a virtual reality environment. Users move between application windows by manipulating their camera-based perspective view of the environment. In Data Mountain, the camera perspective is fixed, but users can still place windows on a 3D plane; this allows windows to be organized at different depths.

Zoomable desktop interfaces, like Pad++ [16], provide users a larger desktop to open more windows. Users navigate through the enlarged desktop by panning and zooming. Time-based representations, like those created by Rekimoto [107], allow users to organize information windows along a timeline of when those application windows are used.

Like the Rooms system, Scalable Fabric [111] allows users to place applications in task related groups. However, Scalable Fabric provides a representation of application groups along the periphery of the user's screen, allowing users to maintain an awareness of those applications not currently being used. Further, this interface allows application windows to easily transition between task groups and for windows from multiple tasks to be in the user's work area at the same time.

While not directly transferable to multiple display environments, there are many design lessons from this body of work that can be applied to multiple display environments. For example, the advantage of spatial layouts in Task Gallery, Data Mountain, and others are carried over into the world-in-miniature representations (Chapter 4). We also utilize zooming techniques in the *detailed view* of our second world-in-miniature prototype (Chapter 4) and in the *shared screen dock* of IMPROMPTU (Chapter 5).

2.8 Evaluations of Co-located Groupware Systems

There have been many informal usability evaluations of systems and tools for supporting co-located collaboration. In [74] the researchers implemented their input redirection system and interaction technique across multiple configurations and preformed a quick needs analysis to determine how well their configuration met different task requirements. When creating the DiamondSpin toolkit, the researchers created multiple applications to demonstrate the toolkit's features. Through informal observations of users working with their prototypes, the researchers gained some anecdotal evidence on the effectiveness of their interfaces and interaction techniques [128].

In each of these studies, no clear experimental goal was formed and no general design lessons were produced. In our work, we perform evaluations of our interfaces and

framework with the goal of producing generalizable design lessons. In doing so, we often compare our interface or framework against alternative designs to provide tangible results showing how our work improves the current state-of-the-art.

Recently, there have been a number of evaluations of tabletop systems. Most of these evaluations [47, 60, 93, 113, 118] explore interactions designed specifically for tabletop systems. However, many of these studies have produced lessons that have applicability to broader domain of digital collaborative workspaces. For example, Morris et. al. found that users benefit from their own set controls rather than a central set of community controls [93]. In [47], Everitt et.al. performed an evaluation of their MultiSpace interface. In the evaluation, the researchers observed users performing two tasks: a image categorization task and a document creation task. From their study, they were able to make some general observations about how users perform collaborative work in a multiple display environments. Among their observations was that users were easily able to transition between parallel and collaborative tasks, and that users utilize devices in different ways depended on the task. In this dissertation, we have used the applicable design lessons from these tabletop evaluations to inform and guide our work.

Within the context of MDEs, there have been fewer empirical studies. Nacenta et al. compared different interaction techniques for interacting across multiple display surfaces in an MDE [96]. In their study, individual users performed a rapid sequence of prescribed relocation tasks. Their results show that users preferred and performed better using interfaces that provided users a visual summary of the workspace (referred to as a radar view in their study).

Streitz et al.'s study compared the effectiveness of different workspace configuration to support collaborative activities [137]. Three configurations were compared: individual workstations only (IW), large shared display only (LD), and individual workstations with large display (IW+LD). The results show that the teams that used individual workstations with the large display (IW+LD) produced better quality work, generate more ideas than in the other two conditions, and also employed a more effective distribution of different cooperation modes.

In this work, we significantly extend our understanding of how MDEs are used to support collaborative activities. Specifically, we report results from one of the first field studies investigating how groups leverage MDEs for performing *authentic* tasks. In this dissertation, we studied how our solution was used to support face-to-face software development. This allowed us to perform an in-depth study to understand how groups use and benefit from MDEs and how supporting frameworks can be improved.

2.9 Emerging Software Development Practices

In this dissertation we concentrated on applying and evaluating our work within a single task domain, group-based software development. We discuss our motivation for selecting this domain in Chapter 4. However, in this chapter we provide necessary background – discussing the importance of collaboration, emerging trends, and existing use of MDE-like workspaces within domain.

Software developers are attempting to improve their work practices to meet the increasing demand for dependable systems [123]. A radical change is occurring in how the act of programming and related development activities is being performed. Development teams are rapidly transitioning from individuals working in their own offices to small groups working face-to-face in co-located workspaces [83, 123].

Very representative of MDEs, these workspaces are typically configured with individual work areas but are also equipped with large displays, whiteboards, and other instruments to foster team communication and awareness. Early evidence suggests that the group activities that are facilitated by these workspaces can reduce defects in software and improve the quality of its overall design [83, 98, 132, 149, 150].

Though being situated within the same workspace allows for increased communication, it also increases the need for more effective groupware tools. For example, groups need to share and interact with each other's task artifacts such as code editor windows, debug windows, and web browsers showing examples. Our work contributes a new framework that supports successful principles of group work within these types of co-located workspaces.

Several tools have been designed or could potentially be adapted for face-to-face group development activities. For example, source code repositories like CVS [2] and SVN [8] can be used to help coordinate access to shared code. However, these systems typically embody formal processes, and do not provide an effective means for informally sharing task artifacts during group development activities.

A file server or e-mail can be used for sharing task artifacts. But, these approaches are not sufficient because they do not allow joint interaction or retain their interaction context when passed between users. Using personal devices, with one connected to a large display, is also insufficient, e.g., artifacts from multiple users cannot be shown in parallel.

Tools have been created for better coordinating activities among programmers. For example, Palanír [119] and Augur [49] provide visualizations of recent actions within a shared code repository. FASTDash [25] extends this awareness to include developers' actions within their local integrated development environments (IDEs). Collaborative IDEs, such as Jazz [33], allow users to see who is working within the shared code, receive updates of their actions, and chat with each other. Our framework can be used to complement many of these tools while also providing additional support for sharing other related information. For example, in addition to shared source code artifacts, FASTDash could be launched from a personal device and placed on a shared display for maintaining awareness of group activity. More generally, utilization of the framework allows the core groupware requirements (summarized in Table 2.1) to be better realized within software development task domain.

Chapter 3

Design and Evaluation of a World-in-Miniature Interface Metaphor

As discussed in Chapter 2, an essential requirement of an MDE is to allow users the ability to easily manage and maintain awareness of information artifacts across the various displays. As a simple example, suppose a group of software developers come together to brainstorm designs for an upcoming project. At the beginning of the meeting, each developer uses the private workspace on his personal device to independently create his own lists of requirements, schematic diagrams, interface sketches, etc. The developers then quickly transition from working individually to working as a group to exchange each others' ideas for review, comment and/or modification. As ideas are narrowed, the developers relocate the designs they like the most to a shared screen in the workspace so that they can more easily review and revise the designs as a group.

This example highlights some of the challenges of realizing the groupware requirements described in Chapter 2 in a multiple display environment. Some of these challenges include enabling *running* applications (not files or links) to be exchanged between devices, supporting relocation of applications between both shared screens *and* portable devices, and enabling users to redirect local input to different screens *without* having to physically move among them.

To address these and other challenges of supporting effective artifact sharing in MDEs, previous work has investigated building interaction techniques that explore the use of textual interfaces [15, 75, 104, 121] and paths that virtually connect devices [74]. Textual interfaces, such as a menu that lists names of devices that correspond to labels attached to those devices, require a user to remember the mappings or to visually scan the space for each relocation or redirection task. Remembering the mappings becomes increasingly difficult as users carry portable devices into and out of the space and as the number of devices increases.

Virtual paths are effective when screens are physically aligned [74], however this approach does not support mobile devices well because their location and orientation may not afford an intuitive path. This approach may also not scale well because as the virtual path becomes longer due to more devices, a user may find it more disorienting to move their cursor or window through the workspace.

To create a more effective interface, we developed a new interface metaphor, called a world-in-miniature interface (or WIM), for managing information artifacts within an MDE (Figure 3.1). The world-in-miniature metaphor enables a user to *visually* relocate artifacts among screens, whether those screens are portable or fixed, small or large, or near or far from the user. With this metaphor, a user can coalesce the tasks of relocating an information artifact and redirecting input or can perform these tasks separately.

In this chapter, we discuss how we iteratively designed our initial interface through a series of low-fidelity prototypes. We describe how we empirically compared the effectiveness of our new metaphor to the current state-of-the-art MDE management interfaces. Using lessons from this comparison, we discuss how we revised our design to improve the scalability and awareness of a world-in-miniature interface without compromising its core benefits. Finally, we describe how we evaluated our revised design within a collaborative task domain and discuss resulting design recommendations.

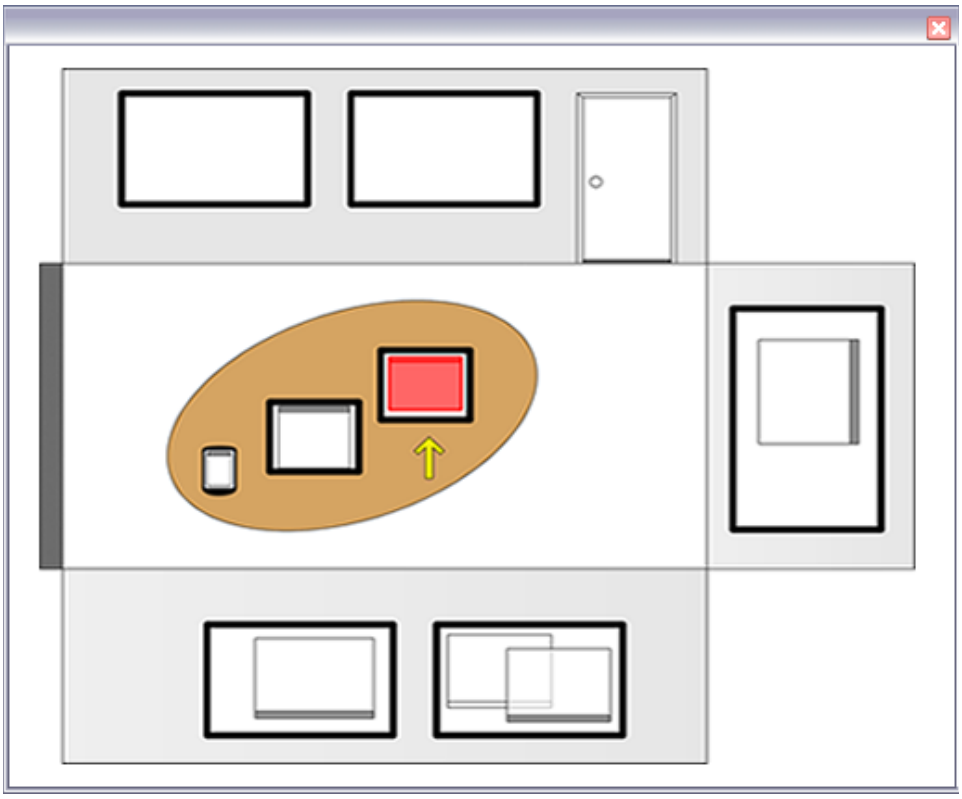


Figure 3.1: A screen shot of the world-in-miniature interface. The map shows each wall that holds a screen as if it had been pulled down on its back side. The orientations of information artifacts are consistent with this model. The oval shape is a table with a PDA and two graphics tablets on it. Compare to Figure 3.2 which shows the physical workspace this interface represents.

3.1 Iterative Design of Our Initial World-in-Miniature Interface

Following a user-centered design process, our first step was to identify user tasks that any MDE management interface should support. We identified tasks by talking with users of an existing MDE workspace and by outlining usage scenarios similar to the example scenario illustrated at the beginning of this chapter. From these efforts, we believe that an effective management interface for an interactive space should enable (at a minimum) a user to:

- Relocate a window from the local (attended to) screen to a different screen in the space.
- Relocate a window from any screen to the local screen.
- Relocate a window between two screens from a different screen (neither of those two) in the space.
- Relocate a window independent of how many screens are physically in-between the source and destination screens.
- Re-position a window on a screen from a different screen.
- Redirect local input to a shared screen as part of the interaction for relocating a window to that screen.
- Redirect local input to a shared screen independent of relocating a window to that screen.

Because a portable device is typically private to a user, we do not include a task to redirect local input to a portable device. In our designs, we assume that applications are just one window, thus we use the terms *application, window* and *information artifact* synonymously.

Using these task requirements, we iteratively designed and evaluated a series of paper-based low-fidelity prototypes. User evaluations took place in our MDE laboratory (see Figure 3.2). As recommended in [16], our evaluation team consisted of a facilitator, note-taker, and "computer." While a user performed tasks with our prototype, the "computer" would physically move to different screens in the space and post storyboards, overlays, or sticky notes to simulate the effects of user actions. Users were instructed to think aloud while performing the tasks. We used the screens in the lab, which were turned off, as a backdrop during the evaluations to better simulate interaction realism in the evaluations.

In the first iteration, we evaluated three very different designs. Based on an evaluation of the designs, for the second design iteration, we selected a specific design, revised it, and then evaluated the revised design. In both evaluations, we identified usability issues through observation of users, analysis of the video and verbal protocol, questionnaire

Figure 3.2: Our multiple display laboratory. Notice how the device layout matches the corresponding part of the world-in-miniature interface in Figure 3.1.

feedback, and user discussion. From this process we uncovered usability issues and learned lessons to inform the design of our first functional interface prototype.

In our evaluations, we had users perform mainly window relocation tasks because we believed those tasks would influence our design the most, however, our functional prototype supports each task identified in the task analysis.

3.1.1 Iteration I - Multiple Low-fidelity Prototypes

In our first iteration, we paper-prototyped three widely different designs:

- *A select and drop interface*. A user selects a button on a window's title bar. Once selected, drop buttons appear on each screen in the space. A user selects the drop button on the desired screen to relocate the window. Part of this design is shown in Figure 3.3a.

- *A world-in-miniature map interface*. A user drags a window back and forth quickly to invoke an world-in-miniature map of the space showing each large screen in the space and provides a drop-down list of the mobile devices. The user taps on a large screen or selects a screen from the list to relocate the window. Part of this design is shown in Figure 3.3b.

- *An egocentric map interface*. This interface was similar to the previous one except that the map was scaled to fit around the desktop, giving a more egocentric feel. Part of this design is shown in Figure 3.3c.

To evaluate the interfaces, we had six users perform window relocation tasks with each interface in the Active Spaces lab. The tasks were to relocate a Notepad window from a laptop to a shared screen, from a shared screen to another shared screen, and from a shared screen to a laptop. By shared screen, we mean a large plasma screen in the space. Although not exhaustive, we felt these tasks represented the more common tasks that a user would perform in the space and that these tasks would influence our design the most.

Following low-fidelity evaluation principles, our goal was not to evaluate specific usability issues of each design, but to gain high-level feedback from users about the different interaction styles and to facilitate new design ideas.

Because we evaluated early designs, we discuss results qualitatively. From the evaluation, we found that:

- Most users disliked the select and drop interface because it required physical movement among screens whereas the other interfaces did not.

- Most users disliked the use of a back and forth gesture to invoke the interfaces. They felt it was not intuitive, may be difficult to remember, and may conflict with normal window manipulations.

Figure 3.3a: Select and drop interface.

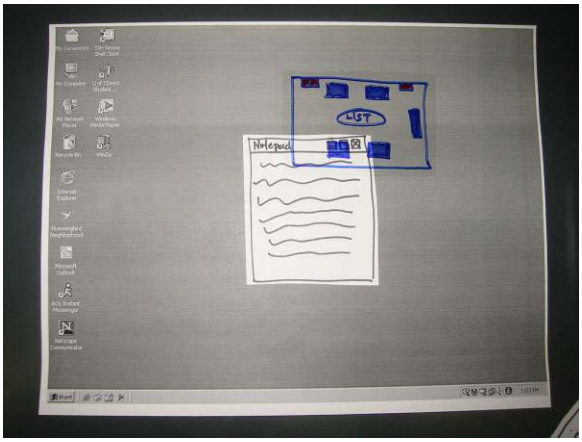


Figure 3.3b: World-in-miniature map interface.

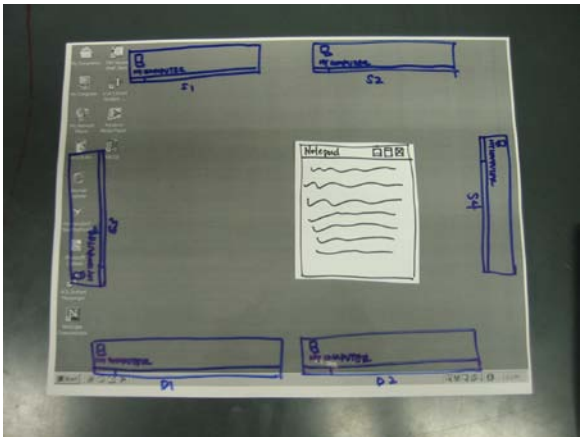


Figure 3.3c: Egocentric map interface.

- Most users disliked the use of a textual list because it was cumbersome to match a name such as “Screen 1” to the corresponding label on a screen in the space.

- Most users liked selecting a button from the title bar to initiate window relocation because they viewed relocation as a window manipulation task.

- Most users liked the world-in-miniature interface designs but wanted stronger orientation cues to better orient the map with the actual space. Users wanted the interface to position the map such that “up” was always in the direction they were facing in the space.

3.1.2 Iteration II - Revised Low-fidelity Prototype

Based on results from the first evaluation and our design discussions, we selected the world-in-miniature interface for further refinement and revised the prototype, see Figure 3.4a. In this revision, we enabled a user to initiate a window relocation task either from the title bar or from a button on the side of the screen (we were not quite convinced yet that an extra button on the title bar was best). Because we conducted a more detailed evaluation of the design, we discuss more details of the evaluation here.

3.1.2.1 Users and Tasks

We had twelve users participate in this evaluation. Most were students in computer science or business. Users performed three tasks based on this scenario:

> *Your manager has asked you to prepare this space for an upcoming meeting. To prepare the space, you need to arrange a presentation, product brochure, and company report on appropriate screens in the space.*

The tasks were to relocate a presentation from a tablet to a shared screen using the tablet, relocate a product brochure from a shared screen to another shared screen using the first screen, and relocate a company report from a shared screen to another shared screen using a PDA. Figure 4.4b shows the evaluation of our revised prototype in progress.

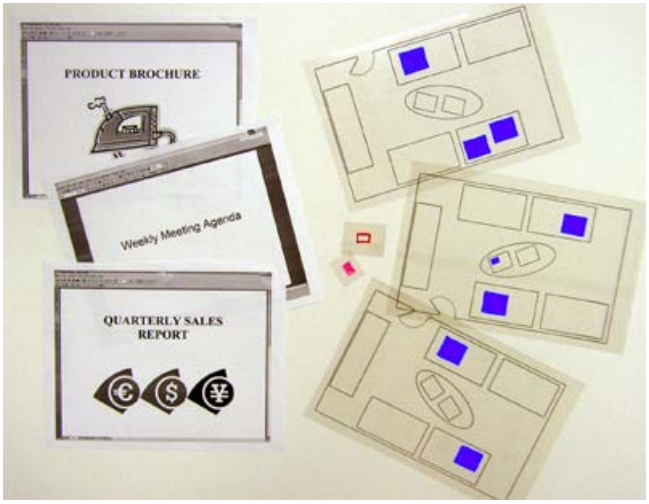


Figure 3.4a: Part of the task and low-fidelity prototype materials used in design iteration II. Task materials included paper-based application windows (left) and low-fidelity prototype materials included transparent overlays (right).

Figure 3.4b. The user (middle) is interacting with the low-fidelity prototype to move an application (in paper form) from this large display to another large display in the space. Based on user actions, the "computer" (right) adds or removes interface screens from the large display, which was used only as a backdrop to enhance realism. The note-taker (left) records usability issues and user comments on a notepad.

3.1.2.2 Procedure and Questionnaires

Upon arriving at the lab, we went through an informed consent process. A user performed the three window relocation tasks in a randomized order. We videotaped each user session for later analysis.

After completing a task, we asked a user to rate how much she disagreed or agreed with these statements:

- The interface enabled you to perform the task easily
- You could repeat the task without help from the researcher (assuming the interface was implemented)
- You could teach someone else how to perform the task using the interface
- You feel the interface was appropriate for the task

After the evaluation, we asked a user to rate how much she disagreed or agreed with these statements:

- You are satisfied with the interface to perform the relocation tasks
- You believe that relocating applications among the screens would enhance a meeting or group activity
- You believe that the use of this space would enhance a meeting or other group activity

For each question, a user responded on a seven-point Likert scale ranging from Strongly Disagree (1) to Strongly Agree (7). We also asked each user to note any specific aspect of the interface that she particularly liked or disliked and if she had any general suggestions to assist in improving our designs.

3.2.2.3 Results and Usability Issues

As shown in Tables 1 and 2, users rated the interface surprisingly high across all dimensions. We attributed the high ratings to users not having an interface to compare

our interface to, users being impressed with the technology-rich space, or users being genuinely satisfied with our interface.

From this second evaluation, we learned that:

Most users disliked the use of a button on the side of a screen to invoke our interface because it was either too far away (large screen) or it just did not seem to fit the task. We are now convinced that a button on a window's title bar is better and used this in our functional prototype.

	Large Screen	**Tablet**	**PDA**
Easy to perform task	6.08 (1.62)	6.58 (.51)	6.33 (.78)
Could repeat the task alone	6.33 (1.50)	6.75 (.45)	6.66 (.49)
Could teach the task to another user	6.25 (1.76)	6.58 (.51)	6.50 (.80)
Interface appropriate for the task	6.08 (1.51)	6.33 (.78)	6.75 (.65)

Table 3.1: Task Questionnaire Responses | Avg (s.d.)

Satisfied with interface for all tasks	6.17 (.72)
The use of multiple screens would enhance a meeting	6.33 (1.07)
Relocating applications among screens would enhance a meeting	6.25 (1.06)

Table 3.2: Post Questionnaire Responses | Avg (s.d.)

Several users were confused by our world-in-miniature map of the space because we did not visually separate the walls from the floor. In our functional prototype, our interface now shows the walls as if they had been pulled down on their back side, but conjoined with the floor.

Several users were confused about the always right side up orientation of applications in our interface because those on the back or side wall should appear inverted or sideways. Our functional prototype corrects this.

Several users had more difficulty orienting the world-in-miniature representations with the space when using the large screens than when using the smaller screens, perhaps because most of the space was behind them when standing in front of a large screen. To add stronger orientation cues, we added an arrow in our representation to communicate a user's location and view direction.

Most users stated that they used physical features in the room such as the door or table to orient the interface's representation with the physical space. We strengthened these features in the world-in-miniature interface in our functional prototype.

Users expected to see immediate visual feedback in the space while interacting with windows in our interface. In our functional prototype, as a user moves the representation of a window in our world-in-miniature interface, it moves a corresponding outline of the window on the physical screens in the space, a technique we call "live outlines."

Most users expected our interface to show which screen they were currently interacting with. In the functional prototype, an the orientation arrow helps to resolve this issue.

3.1.3 Initial Functional Prototype

Based on lessons learned from the user evaluations and design discussions, we developed our first functional prototype interface called ARIS (Application Relocator for an Interactive Space). In Figure 3.5, we show an interaction sequence of a user using ARIS to perform a window relocation task. In this sequence, the user's goal is to relocate an application from the local tablet screen to the shared screen closest to the door.

To launch ARIS and initiate window relocation, the user selects the relocation button on the window's title bar (Figures 3.5a-b). The ARIS interface appears near the user's pointer and the representation of the window from which ARIS was invoked is initially selected (Figure 3.5c). Once invoked, however, a user can relocate *any* application in the interface just by selecting and dragging that window.

A yellow arrow shows the position and orientation of the user in the space. We color the arrow to help prevent the user from confusing it with the screen pointer.

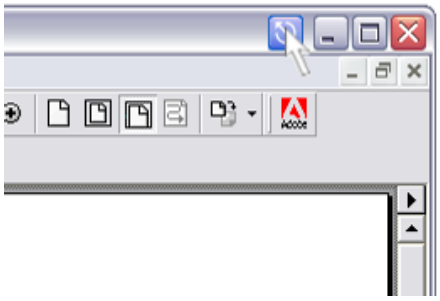
The user then drags the representation of the window to the destination screen (Figure 3.5d). As the user drags the representation of the window across a screen in the world-in-miniature, a “live outline” appears on the corresponding physical screen in the space (Figure 3.5e). This enables the user to look up from the local screen and receive immediate visual feedback of the ongoing interaction. The user positions the representation of the window as desired on the destination screen and then lifts the stylus from the ARIS interface. The application then appears at the specified location on the destination screen (Figure 3.5f), where its border is initially highlighted but fades over a few seconds. The user closes ARIS and the relocation task is complete.

If a user wants to redirect local input as part of relocating an application, e.g., the user wants to continue interacting with the application after it is relocated, the user holds the application still on the destination screen for a short moment (about 2 seconds) in ARIS and then both the application and pointer appear on the destination screen and ARIS closes automatically. To stop input redirection, a user invokes ARIS and holds the pointer over the local screen in the interface for a short moment and then input is redirected. We also provide a hot key to enable a user to quickly end input redirection.

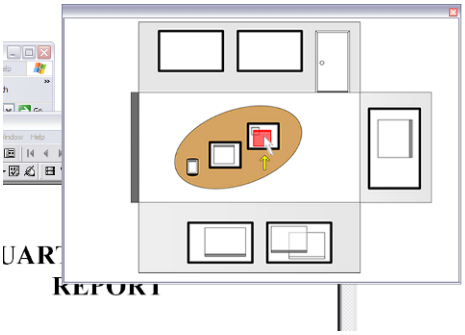
To redirect local input only, a user could position just the pointer over the destination screen in ARIS for a short moment and then the pointer would be redirected to that screen. We based this interaction on the behavior of ‘spring-loaded’ folders on the Macintosh [1]. With a spring-loaded folder, a user holds a file over the folder to open that folder as well as successive folders. Our interface behaves analogously.

As shown in Figure 3.1, the ARIS interface combines many of the successful features explored in our low-fidelity prototypes to create an new interface metaphor we call the world-in-miniature (WIM) metaphor. The metaphor enables a user to quickly establish an overall awareness of the status of the workspace, i.e., to quickly establish which applications are executing on which screens. The metaphor also enables a user to relocate applications visually rather than having to mentally map textual names representing applications and screens to the actual applications and physical screens in the space.

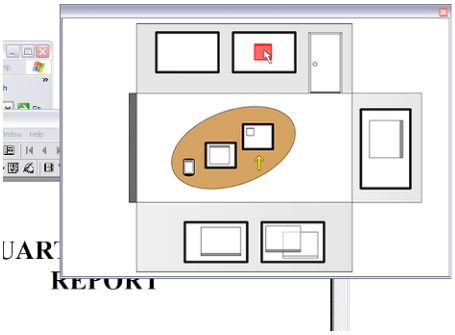
(a) A user wants to relocate an application window to a shared screen in the space.



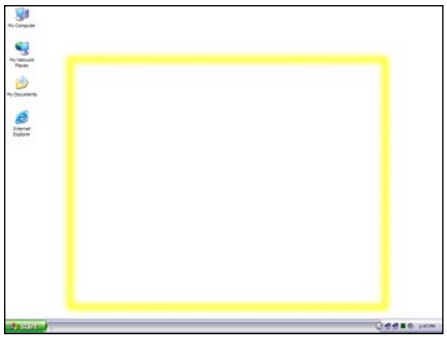
(b) The user selects the button on the window's title bar to invoke ARIS.



(c) ARIS is displayed and the representation of the window from which it was invoked is selected and positioned near the pointer.



(d) The user drags the representation of the window to the destination screen.



(e) A "live outline" of the moving window appears on the destination screen.

(f) The user releases the pointer, closes ARIS, and the relocation task is complete.

Figure 3.5: Interaction sequence to relocate an application window from the tablet to a shared screen using ARIS. The application window moved is an electronic version of the paper document used in our earlier evaluation.

In the interface, we use a semi-transparent rectangle with a thin opaque bar at the top to represent an application window. See Figure 3.5c. The thin bar shows whether a window has focus (darker) and its orientation (bar appears at the top, bottom, or side of the window). When a user selects a window, it gains a semi-transparent red shade. We draw PDAs slightly larger to make them easier to identify and interact with in ARIS.

Using ARIS, a user can relocate applications independent of the distance between the source and destination screens. For example, in Figure 3.5c, a user can relocate the application executing on the screen in the lower left to the screen in the upper right just as quickly as relocating that application between any other screens.

3.1.4 Implementation and Supporting Framework

We developed ARIS using Microsoft Visual C# .Net and Visual C++ to create COM components that integrate with an existing interactive workspace framework called Gaia [116]. We draw the world-in-miniature representation using GDI+ and we place the button in a window's title bar using system hooks to extend window drawing events.

We also created a version of ARIS that runs on a PDA with Windows CE, see Figure 3.6. To build the PDA version of ARIS, we used Microsoft's .Net Compact Framework and the tools supplied in Gaia. Although the world-in-miniature representation and interaction style used in the PDA is consistent with Figure 3.5, the scaling of the map makes the interaction difficult.

In developing ARIS, we built on top of Gaia [18, 19], a middleware operating system that manages resources in multiple device environment. While iROS [10] and Aura [23] also provide support for MDE, Gaia was specifically designed to better support the development of ubiquitous computing and context-aware applications [17].

Gaia supports presence detection for users, devices and services, provides context events and services, and supports an information repository for entities in the space [19]. Most importantly, Gaia provides an application framework [18, 20] to construct or adapt existing applications to execute in the space. The framework supports context and

Figure 3.6: ARIS running on a PDA.

resource adaptation, application mobility, and many other application-level services [18]. By utilizing Gaia's space repository [19] and application framework [18, 20], our interface can retrieve information about the presence of users, devices, and applications in the space. By interacting with the application framework, ARIS can relocate running applications among registered devices in the workspace.

When ARIS requests to relocate an application window, Gaia identifies the same application or an application that provides the same service on the destination machine. Gaia then invokes that application and passes appropriate state information from the previous instance. For example, when relocating an application instance of PowerPoint, Gaia closes the application on the source machine, invokes PowerPoint (or an application providing a similar service) on the destination machine, and passes runtime state such as which slide was being displayed. If Gaia cannot find an application supporting the same service, it reports an error, however in the future, it may be possible to extend Gaia to generate interfaces for unsupported services on the fly [13].

Because Gaia is being extended to support orientation information for users and devices [19], ARIS assumes that this information is already available. Although our interface can

only relocate applications built on Gaia's application framework, we believe our work will help encourage the development of more supported applications in the future.

3.2 Lessons Learned from Designing Initial Prototype

Through the design and implementation of our first world-in-miniature interface, we learned several lessons about how to build a more effective interface for a multiple device environment and how to better conduct evaluations of low-fidelity prototypes in these environments:

- *Enable a user to initiate a window relocation task from a window's title bar*. Users disliked the use of a back and forth motion of a window and the use of a button on the side of a screen to initiate window relocation. Users felt that window relocation was a window manipulation task and therefore expected it to be tied to a window's title bar. As shown in Fig. 3.5b, a user can invoke the interface from a title bar. Once invoked, the interface positions itself such that the invoked-from window is near the user's pointer. This enables the user to quickly continue the initial interaction and drag the window representation to the desired screen.

- *Provide a user with an alternative interaction to invoke the interface*. There are situations where invoking the management interface from the title bar of a window may not work. For example, the user may have difficulty reaching the title bar on a large screen or the user wants to relocate a window between distant screens using a local device. In the latter case, the user may not have a window open on the local device or may find it awkward to invoke the interface from a window other than the 'to-be-relocated' window. To provide an alternative interaction, we placed an icon in the system tray from which the user can also invoke the interface. Other interactions such as a context menu may also work.

- *A 2D world-in-miniature representation of the workspace may be good enough for window relocation tasks*. We debated whether our interface metaphor should

provide a more realistic 3D representation of the workspace to help a user better orient it with the physical workspace. We found, however, that users had few problems orienting the 2D world-in-miniature representation of the workspace as long as "up" in the representation was always in the direction they were facing. Users stated that they almost always used the door in the representation to orient themselves within the workspace. Because our interface shows the walls as if they have been pulled down on their back side, we can show each screen screen-side up. Achieving a similar effect in a 3D representation may be much more difficult and would likely be more difficult for users to comprehend.

- *Map user actions through a world-in-miniature representation of the MDE*. Although we observed a few users wanting to relocate a window directly off the edge of the screen onto another screen, we do not believe that this *virtual path* technique would work well in general. Because users can dynamically add and remove screens from the workspace (PDAs, laptops, etc.), adequately conveying how screens are virtually connected may be difficult. Also, when a user wants to relocate a window between screens on opposite sides of the workspace, he should not have to relocate the window across each screen in-between them. Using our interface, a user can directly relocate a window between any two screens in the workspace in about the same time. Finally, MDEs contain input devices that have direct, 1:1 mapping to the display surface (e.g. tablet PC or large panel touch overlay) and cannot support (without modification) the ability to perform input actions that extend beyond the bounds of one display device.

- *Provide feedback in the space as a user interacts with the interface*. While interacting with our second prototype, several users commented that they expected to see feedback of their ongoing interactions in the physical space. To provide this feedback in ARIS, we use a "live outline" technique where our interface moves an outline of the window in the workspace as the user moves the corresponding window in the interface. See Figure 3.5e.

- *Provide feedback that transcends the end of the relocation task*. After completing a window relocation task, we often observed users visually scanning the workspace to be sure that the relocated window was indeed on the desired screen. To provide this feedback in ARIS while being consistent with the use of a live outline, at the end of a relocation task, we exaggerate the outline with color and then fade it out over a few seconds. Other feedback techniques such as animation trails could also be used.

- *The "computer" in a low-fidelity evaluation needs better tools to help her function more efficiently in a MDE*. As an interaction scenario played out, the "computer" had to physically move to the different screens in the workspace. Because of the large number of screens and their physical separation, the necessary movement time can cause unacceptably slow response times for a user. Most importantly, however, it also unintentionally guides a user's visual attention to the appropriate screen, making it difficult to determine whether feedback from the interface is effectively drawing the user's attention. To alleviate these issues, researchers need an interactive sketching tool that enables the "computer" to sketch low-fidelity components (storyboards, overlays, sticky notes, etc.) and then control the display of the components on the screens in the workspace.

- *Use head turns and physical movement as new metrics by which to evaluate the usability of interfaces in MDEs*. From observing users interact with our prototypes and based on discussions with them, we learned that different interfaces cause different amounts of head turns and physical movements and that users are sensitive to these differences. For example, one user said that poor feedback in our second prototype caused him to "look around the room [many times]." We recommend that designers consider the use of these metrics to complement the use of existing metrics for evaluating interfaces in a MDE.

- *Users believe that the use of a MDE could enhance collaborative work*. As shown in Table 2, users perceived great value in using a MDE for collaborative work. By

using ARIS as an enabling mechanism, we believe that our work can enable richer collaborations among users of a MDE and thus can enable researchers to better evaluate the use of a MDE for collaborative work.

Although our usability lessons are based on evaluations of low-fidelity prototypes, Mack and Nielsen [89] have shown that evaluations of low-fidelity prototypes often identify usability issues similar to those identified with a functional interface.

3.3 Addressing Scalability and Awareness

While our experiences in designing and developing our first world-in-miniature implementation revealed many lessons about designing interfaces for MDEs, it also showed that its usability scales poorly when even a moderate number of applications are shown in the interface. Representations of devices and applications become occluded or are just too small, making it difficult to interact with them, especially when using stylus or touch input. We also found that using only a rectangular outline to represent an application in the interface does not provide adequate awareness of the workspace, which is crucial for effective collaboration [140].

To preserve advantages of WIM interfaces while addressing the issues of scalability and awareness, we designed new interaction techniques and prototyped them within a new WIM implementation called SEAPort (Scalable, Enhanced Awareness, Portal-based interface), shown in Figure 3.7. The interface uses zooming and animation-based interactions to improve scalability and uses application icons (as opposed to just rectangular outlines) and portal views with real-time updates to improve workspace awareness.

3.3.1 Goals and Iterative Design Process

Our goal was to develop a new interface that maintained aspects of our previous design known to work well, e.g., the spatial representation, while addressing two significant challenges:

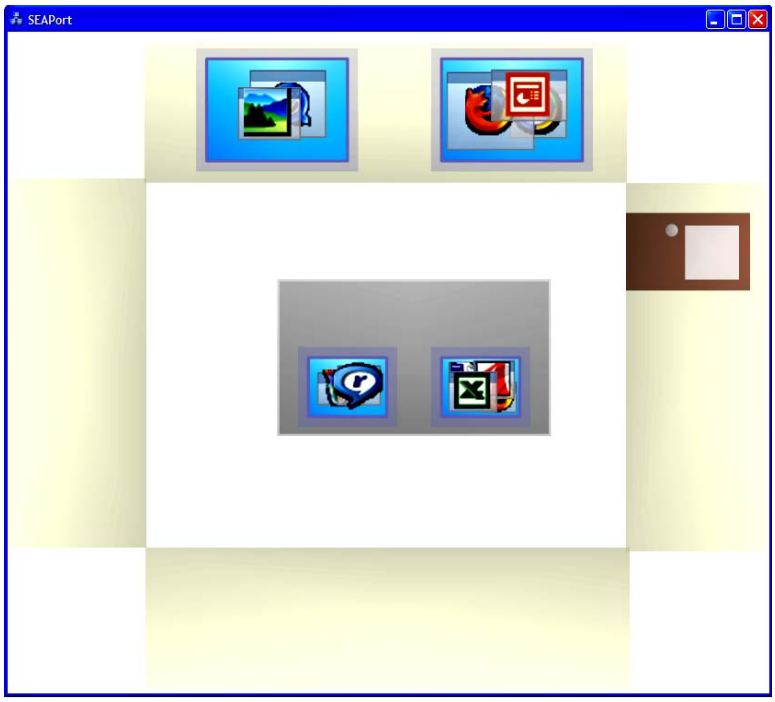


Figure 3.7: The SEAPort interface executing in our MDE. Each device executes an instance of this interface, allowing users to interact with any local or remote content from the local device.

- *Improve scalability*. Users must be able to better interact with representations when they are fully or partially occluded due to the clutter of an active working context, i.e., many applications are open. Scalability also includes being able to interact with small representations of applications when using a mouse, stylus, or touch screen, all of which are common in MDEs.

- *Provide better workspace awareness*. Workspace awareness (WA) refers to the moment-by-moment understanding of the state of a shared task environment and user actions that change it [57]. For example, collaborators need to know what, where, and how task artifacts are being used [57, 140]. For effective WA in an MDE, users need an interface that effectively communicates which task artifacts (applications) are being viewed, which artifacts are on which screens, when a

(a) (b)

Figure 3.8: (a) A screen representation with many applications running, some of which are occluded. (b) The same screen, but now shown in the fan-out view which eliminates the occlusion.

switch is made, etc. These cues are important for co-located groups [140], but absent from most management interfaces for MDEs.

To address these design challenges, we iterated on several paper prototypes. Since we felt that some sort of zooming or animation would be necessary to address scalability, our prototypes focused on exploring these types of visualizations and interaction techniques. We learned that zooming in on a particular screen would work well as long as the context of the surrounding spatial representation was maintained and *not distorted*. Based on this and other lessons, we created a new WIM implementation.

3.3.2 Revised Functional Prototype

Out revised interface prototype provides significant improvements over our initial prototype; (i) it uses an application's icon for its representation in the interface; (ii) it supports a non-occluded, fan-out view of applications on a particular screen; (iii) it allows a zoom-based close-up of a screen while maintaining context *without* distortion; and (iv) when zoomed, it provides a portal view of applications running on that screen with real-time updates. As in the previous prototype, users can quickly redirect input by double-clicking on the representation of the desired destination screen.

For the next sections, assume that a group is working in an MDE, comprised of tablet PCs and large shared displays, to write a research paper. Tasks include interacting with

Figure 3.9: SEAPort with the rightmost screen on the desk shown in close-up view (compare to Figure 3.7). The thumbnails provide a live, detailed view of another screen's contents.

digital artifacts such as related work, graphs, data analysis, Web content, etc. and maintaining an awareness of each others' actions to coordinate the tasks. An instance of the SEAPort interface is executing on each user's device as well as on each shared display in the workspace.

3.3.2.1 Additional Visual Representation of Applications

To cue a stronger mapping between the physical workspace and WIM representation, an application's icon is now used as the representation in the interface (Figure 3.8a). Our decision to use application icons was influenced by [88], which showed that world-in-miniature renderings were effective for identifying applications in a multi-display environment. With a quick glance to the interface, users should be able to better

discriminate which representations correspond to which applications in the physical workspace.

3.3.2.2 Non-Occluded View

When many applications are open on a screen, the desired representation may be occluded, as illustrated in Figure 3.8a. In this case, a user can right click on the source screen's representation to view the applications in a non-occluded, fan-out view (roughly similar to the Mac Exposé interaction). From this view, a user can easily select and relocate the desired application (Figure 3.8b).

Upon selecting the desired representation, the other applications return to their original position. As recommended in [32], we incorporated a short (1 sec) animation to transition between the views to help users understand the change.

3.3.2.3 Zoom-based Close-Up Views

In the WIM interface metaphor, the representation of a screen is shown proportional to the workspace, causing the representation for a smaller screen such as a tablet PC to be small and difficult to select, especially when using stylus or touch input.

To see a larger, close-up view of a screen in the interface, a user performs a *pull-out* gesture by selecting the border of the screen's representation and dragging outward. When initiated, a zooming action centers and increases the size of the selected screen to fill 70% of the interface window (figure 3.9). Testing various fill values showed that 70% strikes an appropriate balance between the ability to view details and preserving the surrounding context.

To relocate an application, the user selects and drags the application's representation while still in the zoomed view. When the cursor crosses the screen representation's boundary, it quickly animates back to its normal view, and the user can complete the relocation *without a mode switch*. Like the non-occluded view, the animation lasts about 1 second and helps reinforce the transition between the interface states.

Since the non-occluded and close-up view interactions are independent, users can invoke either interaction at any time. For example, invoking the non-occluded view in Figure 3.9 would cause applications to fan-out as in Figure 3.8b. For either view, the system manipulates only the local representations, not the corresponding live applications on the physical screens.

3.3.2.4 Portal View

To enable better awareness of the workspace, when a screen is in close-up view, each icon is replaced with a thumbnail of the application, which is updated in real-time. For example, this would be useful to determine who currently has a particular document, but in a *non-intrusive* manner.

3.3.3 Comparative Evaluation of Initial and Revised Prototypes

We compared the scalability and awareness enhancements of our revised prototype, SEAPort, to our previous prototype, ARIS. Since awareness is a complex, multi-dimensional concept, this study focuses on an interface's ability to aid recall of which applications are on which screens in the workspace. Future work will test other awareness dimensions. Our improvements were evaluated as a whole, as they are very inter-dependent and would likely be replicated as a whole, not piecewise in other interfaces.

3.3.3.1 Experimental Design and Configuration

The experiment used a repeated measures design with Interface (ARIS and SEAPort) and Clutter (low and high) as within subject factors and Input Device (stylus and mouse) as a between subjects factor. The presentation order of Interface and selection of Input Device was counter-balanced.

Our workspace reflected those shown in [73, 116]. It consisted of two 61" plasma displays, a 20" LCD screen, and an 18" graphics tablet. The two smaller displays were placed 2' apart on a table in the center of the room. The large screens were placed 1' apart against the wall in front of the table.

3.3.3.2 Users and Experimental Activities

Eight users (4 female) participated in our study and were students in CS or Psychology. Ages ranged from 18-29. To make the experiment as engaging as possible, we used a context that we felt would be familiar and of interest to most users. Users were asked to review and organize content for a multimedia presentation about our school's basketball team.

The first activity evaluated the effectiveness of our scalability enhancements. For this activity, users used each interface to organize content among several screens in the workspace, which consisted of four relocation tasks. The user was given printed instructions, e.g. relocate the document containing rebound statistics to the leftmost large screen. Users performed similar activities in two conditions, one with only a few applications (low clutter) and another with many (high clutter). The latter caused at least half of the representations to have at least some occlusion, but the number of applications open was within practical limits.

Our second activity evaluated how well SEAPort enhanced recall of the state of the workspace. For this activity, applications were configured among the screens, a user reviewed the configuration, and then recalled as much of it as possible, without and then with the aid of the interface. This part of our evaluation was inspired by the Situation Awareness Global Assessment Technique (SAGAT), a commonly used technique for measuring awareness where information screens are blanked out and users are asked to recall task-related information [45, 46].

3.3.3.3 Procedure and Measurements

When a user arrived, the activities were explained, the first interface was demonstrated, and the user practiced using it. The user performed the organizing activity in the low clutter condition (4 applications, 1 per screen) and then performed a similar activity in the high clutter condition (11 applications, 2-4 per screen). The user was asked to perform the activities as quickly as possible. Upon completion, the user filled out a questionnaire.

This process was repeated for the second interface, using a counter-balanced order. Screen capture software was used to record a user's on-screen interaction.

For the awareness activity, the experimenter configured a set of applications among the screens and the user reviewed it for 20 seconds. With screens turned off, the user labeled a printed map of the workspace with the location and content (one descriptive word) of the applications. The screen closest to the user was then turned on showing the interface maximized. Using it as a memory aid (no interaction), the user modified their map as desired. The user performed this activity only in the high clutter condition and the number of applications (11) was well above short-term memory limits [92]. This process was repeated for the second interface.

For the organizational activity, we measured:

- *Completion time*. This was the time from the first cursor movement to the completion of the final task.

- *Error*. We defined an error as any interaction step that did not move a user closer to completing the activity.

- *Subjective feedback*. Users rated an interface across various dimensions, including ease of use, learnability, and overall satisfaction. Questions were taken from [85].

- *Visual scans*. This is when the user shifted visual attention to a screen other than the local screen, an important metric for evaluating interfaces for MDEs [19]. Visual scans were identified by reviewing an over-the-shoulder video of the user.

For the awareness activity, we measured the number of applications correctly labeled on the map in both the free recall and interface assisted conditions. For an application to be marked as correct, both its location and content had to be correct.

3.3.3.4 Results

There were no differences for errors and Input Device had no effect. Thus, these will not be discussed further. An ANOVA showed that Clutter had a main effect on completion time ($F(1,7)=19.07$, $p<0.003$). Completion time was worse for high clutter ($\mu=42.3$s) than low clutter ($\mu=31.9$s), likely due to the increase in choices (Hick's Law). Interface also had a main effect on completion time ($F(1,7)=13.24$, $p<0.008$) with activities being performed faster with SEAPort ($\mu=31.5$s) than with ARIS ($\mu=42.7$s), which represents a 26% improvement. There were no interactions in the data.

Clutter had a main effect on visual scans ($F(1,7)=9.03$, $p<0.020$). There were more scans during high clutter ($\mu=4.6$) than low clutter ($\mu=3.0$), likely due to the larger number of applications. Interface also had a main effect ($F(1,7)=77.54$, $p<0.001$), with SEAPort ($\mu=2.3$) causing fewer visual scans than ARIS ($\mu=5.3$), a 56% improvement.

For subjective feedback, SEAPort was rated higher on five of seven dimensions ($t(7)\geq2.376$, $p<0.049$). Users found the new interface 11% easier to use ($\mu_{SEA}=6.50$, $\mu_{ARIS}=5.88$), 21% more comfortable ($\mu_{SEA}=6.50$, $\mu_{ARIS}=5.38$), 38% better for finding information ($\mu_{SEA}=5.88$, $\mu_{ARIS}=4.25$), and 25% more satisfying ($\mu_{SEA}=6.25$, $\mu_{ARIS}=5.00$) for the tasks.

For the recall data, there was no difference between the baselines ($\mu_{SEA}=7.9$, $\mu_{ARIS}=7.3$), consistent with short-term memory limits. However, users were able to recall much more of the configuration with SEAPort ($\mu=10.8$) than with ARIS ($\mu=7.8$; $t(7)=5.3$, $p<0.001$), a 28% improvement. The revised visual representation in SEAPort thus enables users to better extract which applications are on which screens, an important component of awareness.

Overall, the empirical results confirm that our new interaction techniques have made significant strides towards meeting our goals of improving scalability and enhancing awareness. These techniques advance the use of the WIM metaphor for managing applications in MDEs. Also, the interaction techniques and revised visual representations can be leveraged in other world-in-miniature user interfaces, e.g., [76].

3.4 Comparison of WIM Metaphor to Current State-Of-The-Art

Our next step was to compare our revised WIM design to two other existing, commonly used interfaces for MDEs. Comparing different classes of interfaces would allow us to better understand each interface's strengths and weaknesses, learn which situations make one interface perform better than the other, and learn lessons that can lead to a set of design principles for MDE interfaces. In addition to the WIM interface, two other classes of interfaces that are commonly used in MDEs were selected: textual and map.

The textual class of interfaces provide identifiers for applications and devices in an MDE (e.g., [116, 120]). To relocate applications, users select the applications and the source and destination device, e.g., by selecting IP addresses or labels from text-based UI controls. This interface offers simple interaction, but users must learn and recall how identifiers map to the applications and devices.

In the map class of interfaces (e.g., [104]), users are provided with a strict top-down view of the workspace. Users are able to use their spatial reasoning abilities to identify devices based on their location in the environment. However, users must still map textual identifiers to their corresponding applications.

We chose to study the interfaces within the context of a collaborative activity. A comparative study would provide a more realistic task context and work environment to study the three interface classes. We configured a representative MDE consisting of three tablet PCs and two large displays and asked groups of three users to perform problem solving activities within it (activities described in Section 3.4.5). Users needed to plan the activities, coordinate actions, modify, share and exchange task artifacts (applications), and transition between individual and shared work. Each group performed a similar activity with each of the management interfaces. We measured time to relocate each application, subjective workload, and user satisfaction, as well as observed how groups used the MDE to structure their activities.

In the sub-sections below, we describe how our results showed that users could relocate artifacts faster with the WIM interface and preferred it over the others. We also explain

how the WIM interface may not always be the most effective interface for an MDE. For example, if the location of devices changes often due to users moving around in the workspace with their devices, it would be cumbersome to keep the spatial representation up to date.

Derived from empirical results, observations of how users structured their activities, and an analysis of how users interacted with each interface, we present a set of design lessons for improving MDE management interfaces. We demonstrate the applicability and practicality of these lessons within the world-in-miniature interface, but analogous improvements could be made to the others.

3.4.1 Experimental Design

Our study was designed to answer these questions:

- How does the interface affect relocation performance, subjective workload, and user satisfaction when used during collaborative activities in MDEs?

- How do users structure their activities in MDEs and how does the interface affect that process?

- What are the strengths and weaknesses of alternative interfaces for managing applications during collaborative tasks in MDEs and what lessons can we learn to improve these interfaces?

The experiment used a mixed design with Interface (Text, Map, and WIM) as a within-subjects factor and Activity (Comic strip and Collage) as a between-subjects factor. Eighteen users participated in the study in groups of three. Users consisted of undergraduate and graduate students, and administrative professionals from our institution. Ages ranged from 18 to over 40. Users were compensated with a $5 gift certificate to a local coffee shop for participating. None of the users had prior experience with MDEs.

Figure 3.10: A group of users performing the collage activity in our MDE. Each user is individually searching for images using several open Internet Explorer windows on a tablet PC. When a desirable image is found, the user relocates the shared canvas (shown on the right-most large display) to the local tablet to add the image. The user may then relocate the canvas back to a large display or directly to another user's tablet.

3.4.2 Multi-Display Environment

As shown in Figure 3.10, our MDE consisted of two 61" plasma screens each driven by an independent machine and three Tablet PCs. The tablets were positioned 2' apart on one side of a meeting table. Each user was seated in front of a tablet and used a stylus for input. The plasma screens were positioned behind the table, 1' apart on the same plane, and within the users' field of view. The MDE used in our study was specifically configured to be representative of the broader class of MDEs designed to support small team work groups (about 2-6 users), as exemplified in [73, 116].

In our configuration, only two (as opposed to three) large displays were used to prevent the MDE from having the same number of personal and shared displays, which we felt would generate more interesting patterns of use. Having three users in each group is representative of small team work groups.

3.4.3 Distributed Drawing Canvas

We created a customized distributed drawing canvas to be used for both experimental activities (Figure 3.11). The application supports basic inking and editing commands using stylus or mouse input. Images can be placed on the canvas through a drag and drop interaction. A lightweight service executes on each device in the MDE, allowing the canvas to be relocated. Over a 100Mbps Ethernet, relocation takes less than 250ms.

3.4.4 Interfaces Studied

The three interfaces compared in the study:

- *World-in-Miniature*. Shown in Figure 3.7, the world-in-miniature interface studied was the SEAPort implementation that was discussed in the above sections.

- *Textual*. As shown in Figure 3.12a, the textual interface was composed of selection lists and buttons. To relocate an application, a user selects the source screen, application to relocate, the destination screen, and then selects “Relocate Application.” Selection of the application and devices is made from a list of textual identifiers. For the applications, the identifiers matched the text used in their title bar. For the devices, identifiers matched labels that were attached to the devices, e.g. “Plasma Display 1.”

- *Map*. As shown in Figure 3.12b, the map interface provides a strict top-down view of the MDE. Each device is represented as a thin rectangle with the tablets being shown on the table. The placement of the representations matches the position of the devices within the physical workspace. To relocate an application, a user selects the representation of the source device and a drop-down list appears

showing a textual list of running applications. Identifiers in the list match the text in the applications' title bars. A user selects and drags the desired text item to the representation of the target device, which flashes locally to indicate the action has been completed.

When a specific interface was assigned, an instance of that interface was executed on each user's tablet. This was done to limit the interaction overhead of having to access the interface from a menu or other control. Our experience suggests that this is similar to how the interfaces would be configured and used in practice. Each interface went through at least one round of usability testing prior to the study to ensure a fair comparison.

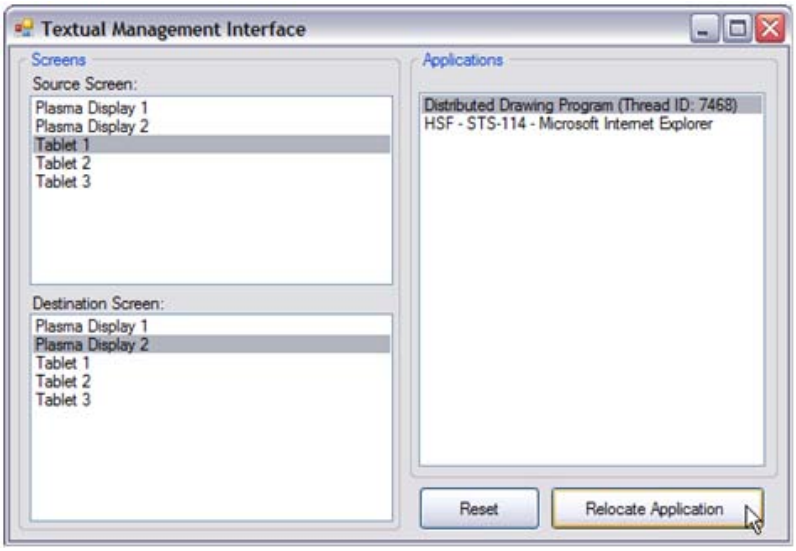
The interfaces chosen typify interaction designs used in many existing MDEs or similar environments. The textual interface typifies interaction designs in MB2Go [75], iROS [72], and Gaia [116], where at least part of the interaction is to select textual identifiers of applications or devices. The map interface typifies interaction designs in iCrafter [73] and Mighty Mouse [27], where the visual representation in the interface reflects the spatial arrangement of the physical devices.

Our study did not include use of a virtual path interface, the extension of a multi-monitor interaction, as it is often not a practical solution for an MDE. Though used in some prior work, e.g., [74, 108, 127, 139], this technique requires the cursor to be controlled beyond the user's local screen, requiring the use of a *relative* input device. Thus, it cannot easily support *absolute* devices such as stylus and touch input, which are prevalent in MDEs. Each interface tested in our study can support mouse, stylus and touch input.

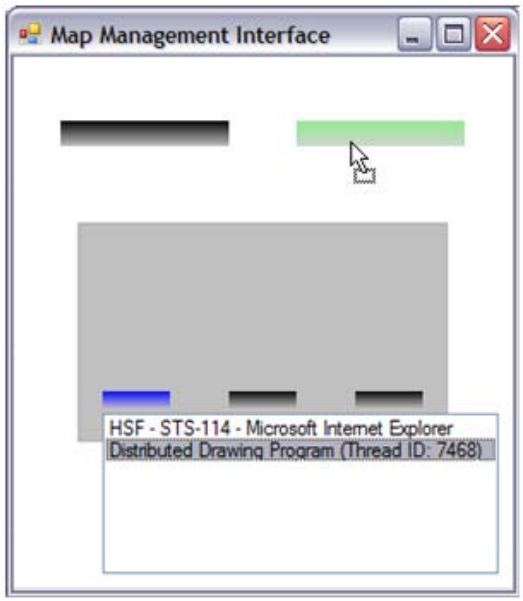
(a)

(b)

Figure 3.11: A sample of task artifacts created by users during the study. 3.11(a) shows a collage created during a collage activity while 3.11(b) shows one frame from a comic strip activity.



(a) Textual Interface



(b) Map Interface

Figure 3.12: The two alternative interfaces used in our study. The interfaces are shown from the perspective a user sitting at the table. Each is currently showing a user in the process of relocating an application from the tablet on the left to the large display on the right.

3.4.5 Collaborative Activities

Two collaborative activities were designed for the study:

- *Create a digital collage*. The purpose of this activity was to produce a meaningful digital collage about a recent news event, e.g., NASA's return to space. Each user's tablet had Internet Explorer running which showed a unique set of links to relevant news sites. Users visited the sites searching for images related to the given theme. When an image was found, the user found and then relocated the shared collage (an instance of the distributed drawing program) to his tablet, dragged the image to the collage, sized and positioned it, and then optionally relocated it to another device. The final collage was to contain at least nine images with each user contributing at least three.

- *Create comic strips*. The purpose of this activity was to sketch two separate comic strips, each with three frames. Each frame corresponded to an instance of the drawing canvas. To seed the group's creative thinking, we provided content for one frame of each strip and asked the group to sketch the other two and add dialog to all three. For example, one given frame had a sketch of a person crossing the street. At the start of the activity, six drawing canvases were executing on the two large screens. To facilitate the need for coordination, each user was asked to choose 1 of 3 responsibilities; creating characters, creating scene content, and adding dialogue. If desired, the group could devise an alternative work plan.

These activities are representative of creative problem solving tasks that small teams of users often engage in and are a central type of activity that MDEs are designed to support. The tasks also required different degrees of coordination. For example, for collage generation, users needed to coordinate access to just one shared resource whereas creating comics required coordinated access to multiple resources. By engaging users in these activities, we were able to study how users leveraged personal and shared devices to exchange digital artifacts, coordinate actions, maintain awareness, transition between

individual and group work, etc. and how each management interface affected those behaviors.

3.4.6 Procedure

When users arrived at the lab, we went through an informed consent process. Users were then trained on the drawing tool, given general instructions for the assigned activity, and introduced to the first interface. Three groups (each with three users) performed the collage activity and three groups performed the comic strip activity. The assigned activity was performed once with each interface. Interfaces were presented using a Latin Square design.

A group was given 15-20 minutes to perform the activity and then completed a subjective workload and interface questionnaire. This process was repeated two more times for the other interfaces. Subsequent activities were varied slightly, e.g., a different theme and web sites were provided for collage generation while different seed sketches were provided for the comic strip activity.

After using the last interface, the experimenter led a group discussion about the use of all 3 interfaces. Camtasia was used to record users' screen interaction and the entire session was videotaped.

3.4.7 Measurements

For both activities, we measured:

- *Time to relocate an application*. This was measured from when a user made a directed action with the interface to when the application appeared on the destination screen. Measurements were computed from time stamps in the Camtasia videos.

- *Subjective workload*. This was measured using the NASA TLX [62], which measures workload along 6 dimensions: *mental demand*, *physical demand*,

temporal demand, *own performance*, *effort*, and *frustration*. Responses are marked by drawing a vertical line along a continuous scale from Low to High.

- *Satisfaction*. Users rated each interface according to simplicity, comfort, awareness, and satisfaction. Ratings were structured using a 7-point Likert scale (7 was most positive).

These measures are representative of those used in prior work [19, 96]. In addition to these quantitative measures, we also observed how users interacted with the MDE, the interfaces, and each other during the collaborative tasks.

3.4.8 Results

We discuss qualitative results derived from observing how users structured their activities in the MDE, open-ended responses from questionnaires, and group discussion. Then, quantitative results for performance, subjective workload, and user satisfaction are compared among the interfaces.

3.4.8.1 Qualitative Results

In this section, we discuss how groups structured their activities, detailing how work was divided, how personal and shared devices were utilized, and how users coordinated individual efforts. We then discuss salient qualitative differences among the interfaces.

All groups were able to successfully complete the activities using each interface in the allotted time. This shows that groups can meaningfully collaborate in MDEs. Several users noted that these types of environments seemed well suited for collaborative work, e.g., one user stated “I can definitely see this workspace being used for this type of task.” Another added that she would want to work on her group-based class projects in this environment.

When starting an activity, groups would devise an overall plan and discuss how to divide the work. For example, when creating comics, users would develop a shared vision for the strips and decide who would draw which parts. For the collage, planning entailed

determining an initial order for passing the collage (though rarely followed). Periods of individual work and group discussion were seamlessly interwoven during the activities, showing that the use of MDEs can effectively support this common and important component of collaborative work [140].

Groups leveraged the large displays to provide a shared workspace to discuss intermediate outcomes and organize workflow. For example, once a user finished their part of a comic's frame, they would relocate it to a large display, position it on the screen so that it was closest to the person they thought needed it next (if possible with the interface), and tersely announced its availability. At the end of the activity, groups typically used the large displays to organize the final artifacts. Likewise, when creating the collage, users would sometimes relocate it to a shared display to show the group an interesting image or to discuss if the selected image was an appropriate fit for the given theme.

Users would also exchange task artifacts directly between their personal devices. For example, during the collage activity, a user would add an image to the collage and then keep it locally. When another user wanted it, they would call out whether they could have it. If so, the current owner would relocate the collage to the requesting user's device (place). In a few cases, a user would relocate the collage from another user's device to his local device (take), but this was generally considered socially inappropriate.

One difference that affected group behavior was the amount of workspace awareness (which artifacts are on which screens) that was immediately visible in each interface. When using interfaces with less awareness information (e.g., map and textual interfaces), users compensated by verbalizing more awareness updates (e.g., "I am taking frame 3 now") and requests (e.g., "who has the collage?"). Most users found these verbal broadcasts to be disruptive to their individual work. This finding indicates that it is necessary to visualize an adequate level of workspace awareness in a management interface, despite the fact that users are co-located. This was one reason why users preferred the world-in-miniature interface, as they could extract awareness information with a quick glance.

Another qualitative difference was in how the interfaces affected a user's perception of the workspace. For example, users commented that with the WIM interface, the workspace seemed more cohesive than when using the other interfaces and that the WIM interface made it much more "inviting" to use the shared displays. This is important since a driving motivation for using MDEs is to facilitate sharing of information among group members.

3.4.8.2 Quantitative Results

A MANOVA showed that Interface did not affect ratings of subjective workload, though the trends favored the world-in-miniature interface (36.9%, 46.2%, and 42.8% of maximum workload for the WIM, map, and textual interfaces, respectively). This result shows that users did not find any one interface to be substantially more demanding to use than the others.

An ANOVA showed that the interface had a main effect on how quickly users could relocate applications ($F(2,324)=51.64$, $p<0.001$). Post hoc analysis showed that a user relocated applications faster with the WIM interface ($\mu=2.52s$, $sd=0.63$) than the map ($\mu=5.78s$, $sd=0.47$; $p<0.001$) and textual interface ($\mu=12.12s$, $sd=0.71$; $p<0.001$). The WIM interface thus provides a meaningful performance improvement over the map and textual interfaces (~56% and ~79%, respectively). Users could also perform relocations faster with the map interface than the textual interface ($p<0.001$), which is also a meaningful improvement (~52%).

The difference between relocation times for each interface is only a few seconds. This may be of little concern if relocations are performed infrequently. But, when a group is deeply engaged in creative problem solving, they would want and need to frequently exchange artifacts among devices as quickly as possible (e.g., when debating alternative outlines for the results section of a paper). In these cases, the cumulative effect of these small differences could severely inhibit the free and rapid exchange of alternative ideas, which is crucial during the creative process [134].

An action analysis [84] showed that the differences in performance cannot be explained simply by differences in the number of operators, as each interface required about the same number of steps (4-5). We attribute the differences to "think" time, where users are bridging the semantic gap between the state of the workspace (which applications are where) and the representation in the management interface. For example, when performing relocation tasks in the map and textual interfaces, analysis of the videos showed that users would pause for a few seconds to determine which applications/devices mapped to which identifiers, while their interaction was more fluid with the WIM interface.

An ANOVA showed that Interface affected ratings of simplicity ($F(2,34)=4.46$, $p<0.019$), comfort ($F(2,34)=4.24$, $p<0.023$), awareness ($F(2,34)=6.42$, $p<0.004$), and satisfaction ($F(2,34)=4.65$, $p<0.016$). Post hoc analysis showed that users rated the WIM interface higher along each dimension ($\mu=5.89$, 6.00, 5.44, 5.18, respectively) than the map interface ($\mu=4.94$, 5.06, 4.28, 3.78; $p<0.02$, $p<0.03$, $p<0.05$, $p<0.03$, respectively). Users were also more satisfied with the WIM interface ($u=5.18$) than the textual interface ($\mu=4.17$, $p<0.05$). No other differences were detected. Results indicate that users had a reasonably strong preference for the WIM interface over the other interfaces.

3.4.9 Discussion

Results showed that, compared to the other interfaces, the WIM interface enabled faster relocation of applications and better awareness of the workspace without inducing a measurable increase in workload. The WIM interface was also the interface most preferred by users. The differences between interfaces are likely due to the WIM interface providing a more comprehensive spatial and visual representation of the workspace, allowing it to be more efficiently processed [146].

To be effective, an WIM interface must obtain information about the spatial arrangement of devices in the workspace. This becomes especially difficult when devices participate only briefly in the MDE or participate after the initial workspace representation has been defined. One solution is to leverage our existing configuration tool which integrates with the interface runtime to allow the spatial representation to be dynamically constructed

and modified on the fly. For example, just before the start of a collaborative activity, the tool could be used to configure the spatial layout of participating devices within the interface. Another solution, similar to [127], is for a device to connect to an established MDE session. Once connected, a representation of the device would appear in the WIM interface and could then be manually positioned.

Though we believe that WIM interfaces would be effective for many MDE configurations and group activities, it may not always be the most effective interface. For example, if the locations of devices are constantly changing due to users moving around in the workspace with the devices, it would be cumbersome to keep the spatial representation up to date. Likewise, if users wanted to manage applications on a device with limited display size, such as a PDA, the WIM interface would be less effective.

Thus, in the next sub-section, we discuss design lessons that can be generally applied to the class of MDE interfaces. These lessons were derived from our empirical results, observations of how users structured their activities in the MDE, and an analysis of how users interacted with each interface. We then describe how the lessons can be implemented within the world-in-miniature interface.

3.4.10 Lessons for Management Interfaces

From the study, we learned the following lessons about how to better design better management interfaces for MDEs:

L1. Provide a view that allows all applications to be seen at once. When using the textual and map interfaces, users often explored each device's content in search of a specific application. Several users commented that they wanted to be able to glance at any device and know what was running on it, as they could with the WIM interface. For example, one user stated "the WIM interface was nice because it allowed you to see everything at once." Other interfaces could offer analogous design features. For example, the map interface could have an interaction which toggles the drop-down lists of applications for all the devices at once, allowing a more holistic view of the workspace.

L2. Allow users to spatially position the application on the destination screen. The textual and map interfaces placed a relocated application in the middle of the screen on the destination device. Almost every user expressed a desire to control where on that screen the application would appear, pointing out that this was a notable feature of the WIM interface. The other interfaces could implement an analogous interaction, e.g., they could enable a quadrant of the destination screen to be selected or could show a small representation of a device's screen and allow a user to position an outline within it. Such an interaction would address the lesson, while still maintaining the interface's basic metaphor.

L3. Allow an application running on a personal device to show a mirrored copy (shadow) on a shared display. The granularity of coordinating at the application level in MDEs can be too large. Users often wanted moment-by-moment awareness of what other users were doing so they could coordinate their own actions and creative thinking. For example, during the comic strip activity, a user would periodically become "stuck" because they did not know what else to draw until they saw what another user had created. This often required the user to wait for related artifacts to be relocated to a shared display. This delay was often too long. Users expressed a desire to create a shadow of an application to be shown on one of the shared displays. The shadow would allow a user's moment-by-moment interaction with the application on his personal device to be visible to the group. But, the application could only be controlled by the owner of the local device.

L4. Allow users to assign ownership or role-based identifiers to representations of devices. At the start of an activity, groups would devise a work plan and agree upon how screens would be used and the roles that users would fulfill. Since this information had to be retained in short term memory, it was often forgotten and had to be periodically reacquired. User comments reflected the inability to relate devices to the context of the task, as one user stated "identifiers were missing that personal identification." One solution would be to allow groups to externalize part of the task context into the interface itself. For example, in the map or WIM interface, users could configure a label for each screen, such as "Comic 1" for the left shared display or "Character design" for the personal device (i.e., user) assigned to fulfill that role.

L5. Provide feedback in the interface of ongoing interactions of other users. Especially during the collage activity, we often observed two or more users attempting to relocate the same application at the same time, but to different destinations. In these situations, the user who completed the interaction first had his relocation performed, while the others were left confused about where it went. Users shouted "where did it go" or "who took the frame." This shows that although users are co-located, management interfaces need to provide feedback of other users' ongoing interactions, similar to *feedthrough* in distributed groupware [42]. For example, when a user drags an application's representation, other users' interfaces could highlight it with a user-identifying color.

L6. Allow control over whether applications will appear in other users' interfaces. Though each application was part of the shared activity in our study, several users raised concerns about privacy when using the MDE for other activities. Specifically, they were concerned about situations when they would not want other users to be able to see which applications they were running. Even the limited information provided in the interfaces, such as an application name or icon, may divulge too much information (e.g., "Outlook" easily gives away the user is reading email). Users should be able to control whether applications appear in a management interface and how much detail is shown (e.g., which of an application's name, icon, and thumbnail can be shown).

*L7. Allow users to **place** applications onto, but not **take** applications from other users' personal devices*. During the activities, users would occasionally relocate (*take*) applications from another user's personal device to their own. Users emphatically disliked having an application taken from them, even if they were not currently using it, unless permission was given. Users stated that having an application taken was "annoying" and an "invasion" of their personal space. However, users stated that having an application *placed* onto their personal device was acceptable, as long it appeared behind the focus application (next lesson). Similar to a technique first demonstrated in Dynamo [68], users can prevent other users taking an application by "pinning" the application to the local screen, which would not allow other users to relocate it using their interface. Users further stated that "pinned" should be the default setting for applications running on personal devices.

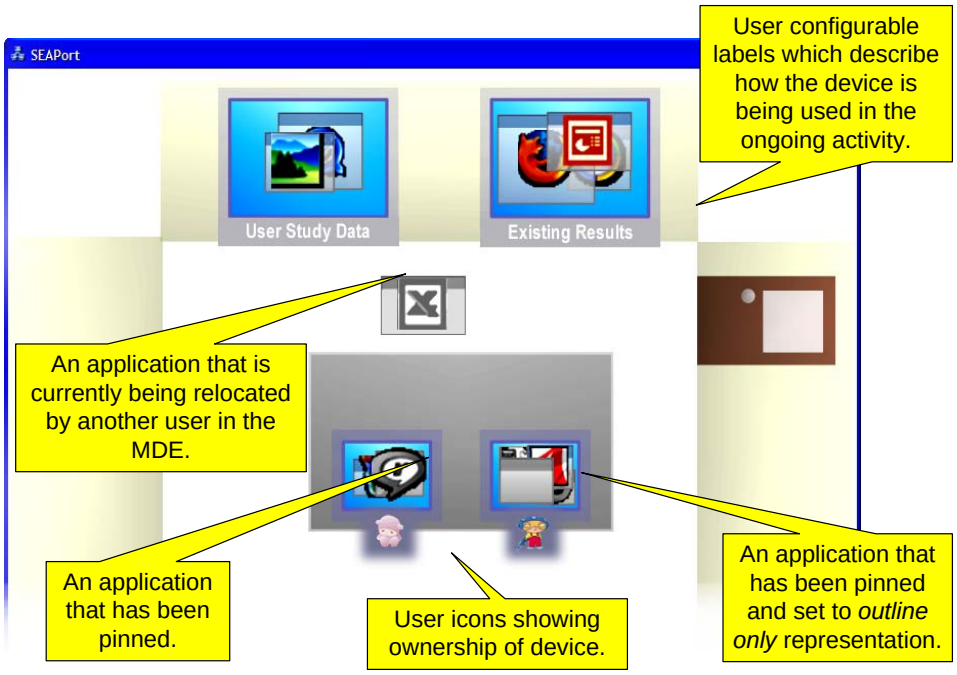


Figure 3.13: The world-in-miniature interface with callouts explaining the improvements derived from our design lessons.

L8. Do not position the application that was relocated in front of the focus application on the destination screen. In each interface, when an application was relocated, it was placed in front of the existing applications. This seemed reasonable when designing the interfaces as it mimics current practice within MDEs, but users were frustrated when an application suddenly appeared and disrupted their current work. Consistent with *L2*, part of the solution is to allow users to position the application off to the side of the destination screen such that it does not interfere with the focus application. However, this is not always possible due to limitations of screen space and application size. Thus, the solution should also include setting the z-order of the in-transit application such that it appears *behind* the focus application.

L9. Provide enough awareness in the interface such that users do not need to compensate with verbal protocols. When using the textual or map interfaces, users often searched devices looking for a specific application or would ask group members where it was. The

latter interrupted ongoing work, which many users found to be annoying. These types of inquiries occurred less often with the WIM interface. This illustrates that communicating workspace awareness (which artifacts are on which devices [57]) in interfaces for MDEs is necessary, despite the fact that users are co-located.

3.4.11 Improving Management Interfaces

We next demonstrate how most of our lessons were used to improve the WIM metaphor. We chose this interface metaphor for improvement since it was shown to be effective.

We addressed L4-8 by adapting our current WIM prototype. This interface already supports the first two design lessons while the third remains work in progress, as it requires building support into the underlying application and systems software. This support could be provided by integrating functionality similar to WinCuts [138]. L9 is implicitly addressed through the others. The interface solutions described next are just a subset of those possible.

(L4) For this lesson, we modified the prototype to allow users to configure labels or icons to the devices (see Figure 3.13). The icons allow users to personalize how their device appears in the interface. Users can also label devices to express the specific role of that device relative to the task or use a label for a personal device in place of an icon. Because the labels do not alter or obscure items within the interface, their addition does not lessen the affordances of the visual representation provided.

(L5) To provide feedback of ongoing interactions, the interface now mirrors other users' relocation actions. As shown in Figure 3.13, to indicate that the application (MS Excel) is being relocated by another user, the application's representation in the local interface appears grayed out and is shown moving between devices. By showing actions *as* they occur, we provide better awareness of the workspace and also prevent conflicting actions between users.

(L6-L7) We hooked into Windows to add two additional buttons to a window's title bar (Figure 3.14). The pushpin allows users to toggle whether other users can relocate the

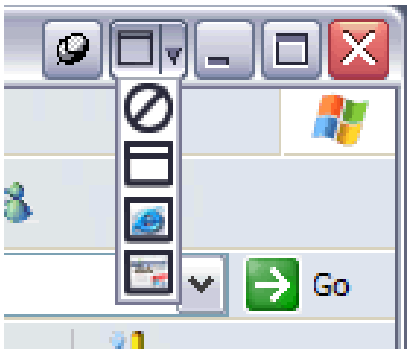


Figure 3.14: Two additional buttons appear on the title bar. The leftmost button allows users to "pin" an application so that other users cannot relocate it. The adjacent button allows the user to specify how the application is represented in other users' interfaces. The options are invisible, outline only, icon, and thumbnail views.

application from their personal device. In the pinned position, the application cannot be relocated, but can be relocated in the un-pinned position. The local user is always able to relocate the application. When an application is placed on a shared device, the pushpin disappears.

The second button allows users to control the privacy of local applications. When a user clicks this button, a drop down menu appears. The menu presents four options for defining how an application is represented in other users' interfaces. The options are; *invisible* (no representation is shown), *outline* (a rectangular outline is shown), *icon* (the application's icon is shown), and *thumbnail* (icons are shown and live thumbnail views are permissible).

(L8) This lesson was addressed by changing the z-order of the application to the top level minus 1 as soon as it appears on the destination screen. Thus, if the in-transit application is placed such that it overlaps with the application in focus, it will appear just behind the one in focus.

These lessons facilitated novel design improvements to the WIM interface – improvements that would probably not have been considered otherwise and are not

directly available from the existing literature on MDEs. Though we illustrated how the lessons could be applied to the WIM interface, they are just as applicable to the other interfaces studied and broader set of interfaces for MDEs. These lessons contribute further understanding of how to design effective management interfaces for multi-device environments.

In this chapter we made a number of contributions that made the concept of WIM interfaces useful for MDEs. However, a limitation of the work so far is its grounding in relatively simple tasks. A remaining challenge is to understand how the WIM metaphor may need to be extended or further revised when confronted with process constraints imposed by an authentic task domain and/or users' preference in that domain. In the next chapter, we discuss how we explored the design requirements for supporting collaboration in one particular task domain, group-based software development.

Chapter 4

Contextual Inquiry to Investigate the Use of MDEs for Authentic Activities

In the previous chapter, we described the user-centered design process we employed to design, implement, and evaluate a new interface metaphor for facilitating effective use of digital information artifacts within MDEs. As a next step, we wanted to understand how MDEs could be applied to support collaborative activities within authentic task domains. Such a transition represents a fundamental shift from focusing on supporting basic interactions (e.g. relocating an application between devices) to supporting effective collaborative practice (e.g. allowing a group to easy review and assess different ideas during a brainstorming meeting).

To accomplish the transition, our design process included several important steps. First, a target domain had to be carefully selected. The domain selected needed to be fairly representative of the broader class of MDE user. Second, once a domain was selected, it would be necessary for us to further develop an understanding of the core interface and system requirements relevant to MDEs. Finally, we had to understand how the requirements of the domain could be best mapped onto and existing framework, or whether a new framework would be necessary.

In this chapter, we begin by describing the selected task domain and the rationale for its selection. Then we describe the user-centered design process for understanding the work practice of the domain relevant for MDEs. To accomplish this, we worked directly with users to study their existing collaborative practices, learn what technologies they

currently use to support these practices, and seek feedback on how they would leverage an MDE to improve their collaborative work.

4.1 Task Domain

Our target task domain for this inquiry was group-based software development. We chose this domain for four reasons. First, software development is a very collaborative activity and requires a high degree of interaction between developers (for a detail discussion see [34, 37, 40, 54, 63, 65, 80, 87, 100, 112, 123, 126, 149, 150]). Second, when developers collaborate, it often involves the use of electronic information artifacts (e.g. source code, documentation, or executing programs). Third, many software developers currently work in co-located configurations in which large displays are present and are likely to adopt MDEs. And finally, the types of collaborative tasks that software developers perform are more broadly representative of the collaborative practices of knowledge workers as a whole.

Improving software development could, by itself, have high value given that it is a $394 billion global industry comprising of over 2.5 million professional software engineers [39]. Improving the known benefits of collaboration in this domain can have significant impact; ranging from improving worker satisfaction to creating more efficient and reliable software [31, 40, 63, 83, 87, 100, 123, 148].

4.2 Surveys and Interviews

We conducted a contextual inquiry that consisted of a series of surveys and interviews with existing and potential MDE users. The purpose of these surveys and interviews was to better understand how MDEs, or partial compositions of MDEs are currently used to support collaboration, what tasks are currently performed in these environments, what functionality is still lacking, and what types of tasks an MDE would be effective at supporting if the additional functionality was provided.

Question	Response Type
How often are co-located collaborative activities performed? (Within the task domain surveyed, these activities include co-authoring, debugging, documenting, brainstorming about code and code reviews. See [82] for further details and definitions)	Multiple choice
What is the breakdown of the different collaborative activities performed?	Table fill in
Compared to team practices one year ago, how has the amount of collaborative activities changed?	Multiple choice
When engaged in collaborative activities, how often are large, shared displays (e.g. a projected display or large plasma panel) utilized?	Multiple choice
What types of information (e.g. applications) are placed on the shared displays?	Free form
How are the shared displays utilized to support the collaborative activity?	Free form
When engaged in collaborative activities, how often do you typically use a personal device (e.g. laptops or tablets) to perform private activities or interact with private information.	Multiple choice
What types of information (e.g. applications) are placed on the personal devices?	Free form
How are the personal devices utilized to support the collaborative activity?	Free form
Please explain how utilizing both the personal devices and shared displays together aids or strengthens your collaborative activities.	Free form
Please explain how you think utilizing the personal devices and shared displays could be improved (either through different/improved hardware or software)?	Free form

Table 4.1: A summary of the questions asked in our survey of current collaborative activities in MDEs.

4.2.1 Methodology and Procedure

Our inquiry was conducted in two parts. We first conducted an online survey which sought to provide an understanding of the broad range of collaborative practices and tools currently in use. Our survey was distributed to developers at Microsoft Corporation, a large software development company. 300 developers were randomly selected from the corporate directory and 90 developers responded to the survey. Participants received an email with a link to the survey posted on the corporate intranet. Table 4.1 provides a summary of the survey questions.

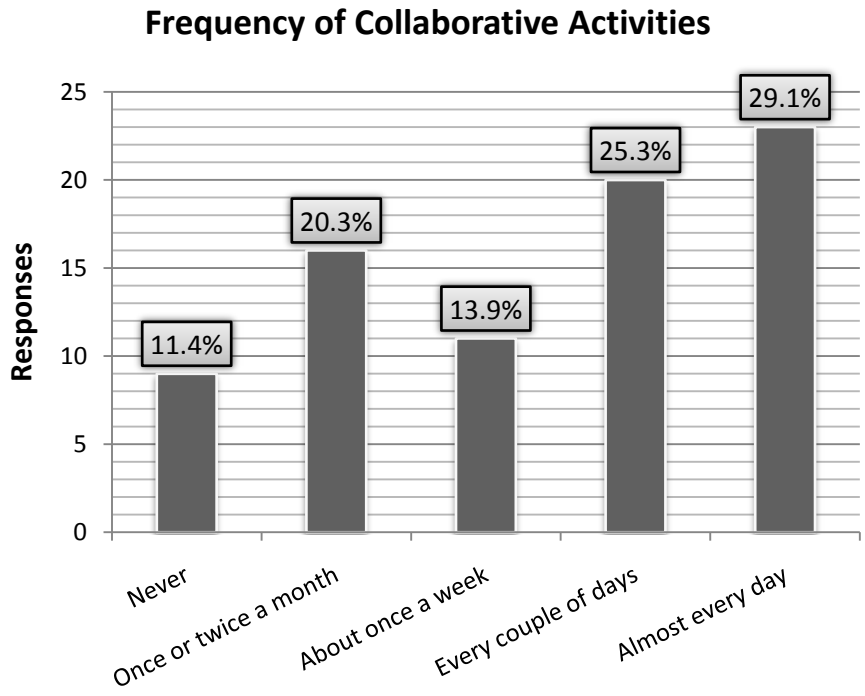


Figure 4.1: A breakdown of the frequency developers perform group-based programming activities. The y-axis is the number of responses per category. Each bar also shows the percentage of overall responses in that category.

To provide more contextual depth to the survey results, we also randomly solicited 13 survey respondents for an hour long, one-on-one follow-up interview. In these interviews, we asked the developers to walk us through a recent collaborative task, demonstrate the tools they use to support their collaborations, and explain how key pieces of information is shared or exchanged. To make demonstrating and explaining easier, interviews were conducted in the developer's workspace. We scheduled most interviews within 72 hours of the developers submitting his survey answers so that his answers would still be familiar and could be used to guide discussion.

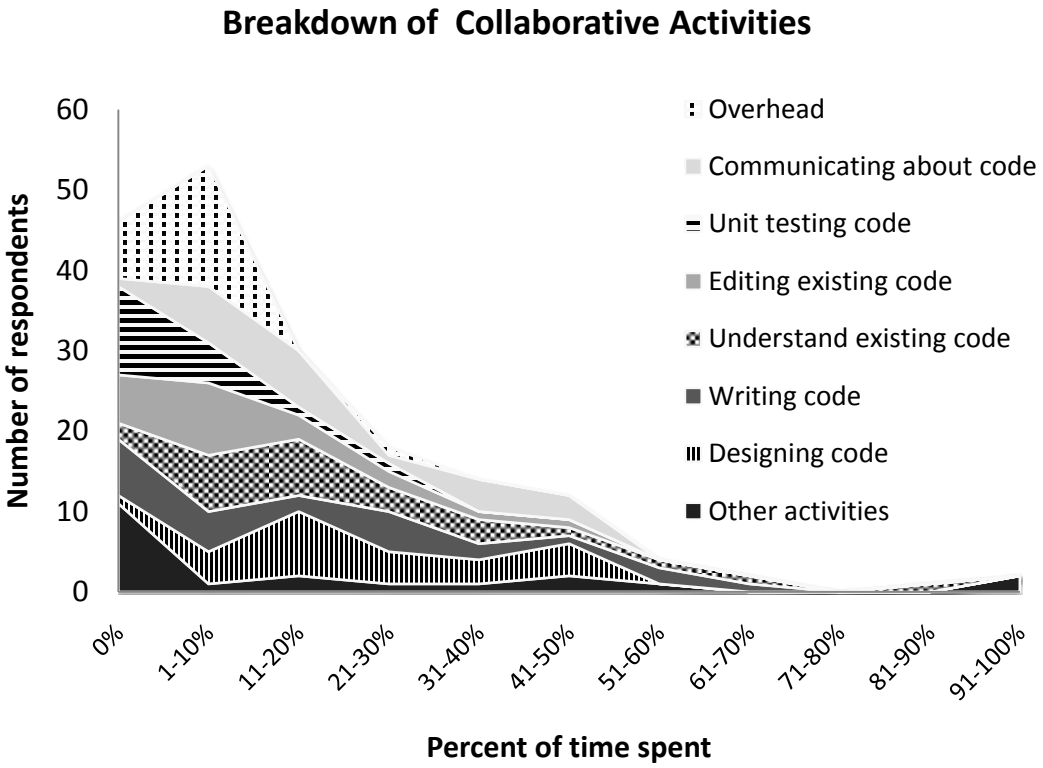


Figure 4.2: A stacked area chart shows the breakdown of the different activities developers said they perform when engaged in group-based development activities. The x-axis is the percent of total group time that is spent; the y-axis represents the total number of responses per percentage. From the graph it can be seen that no one activity dominated the collaborative sessions. Also of note is that the three activities of designing code, understanding code, and communicating about code were the most common.

4.2.2 Survey Results

Though varied in frequency, developers overwhelming indicated that co-located collaborative activities regularly take place within their team. Figure 4.1 shows a breakdown of the 79 responses received for this question. In all, over 68% of the

developers surveyed indicated that they engage in a collaborative activity at least once a week.

The type of collaborations that developers perform is quite varied, as shown in Figure 4.2. There was no particular activity which was dominant across the group of developers surveyed – activities seemed to be equally distributed among the domain specific categories of communicating about code, testing code, editing code, understanding code, writing code and designing code.

Developer responses also indicated a trend towards the frequency of co-located collaborative activities increasing. 38.7% percent of respondents indicated their collaborative activities have increased compared to one year ago, with 16.1% indicating a *significant* increase. 48.3% indicated their level of collaborative activity is about the same. Only 12.9% of respondents indicated a decrease in collaborative activity.

4.2.2.1 Use of Shared Large Displays

To support these more frequent collaboration, users often leveraged a shared large display (e.g. a projected screen or large plasma display) to share information with collaborators. Figure 4.3 shows the breakdown of the frequency of use for the large displays. Over 70% of respondents indicated that a shared large display is a resource utilized in their group meetings. Thus, over 70% of the users in the target domain we studied are already using some form of multiple-display environment.

Figure 4.4 breaks down the types of information artifacts developers place on large displays. This data was collected via a free-form question on the survey and later coded to quantify the responses. Our coding procedure consisted of a researcher creating a list of information artifact classifications based on the survey responses. The researcher then went back and classified each response to one of the classifications. For example, if a survey response indicated that a developer puts an instance of Visual Studio on the large display, this was categorized as an information type of "Code." While our classification scheme is likely not completely inclusive of *all* the different types of information that

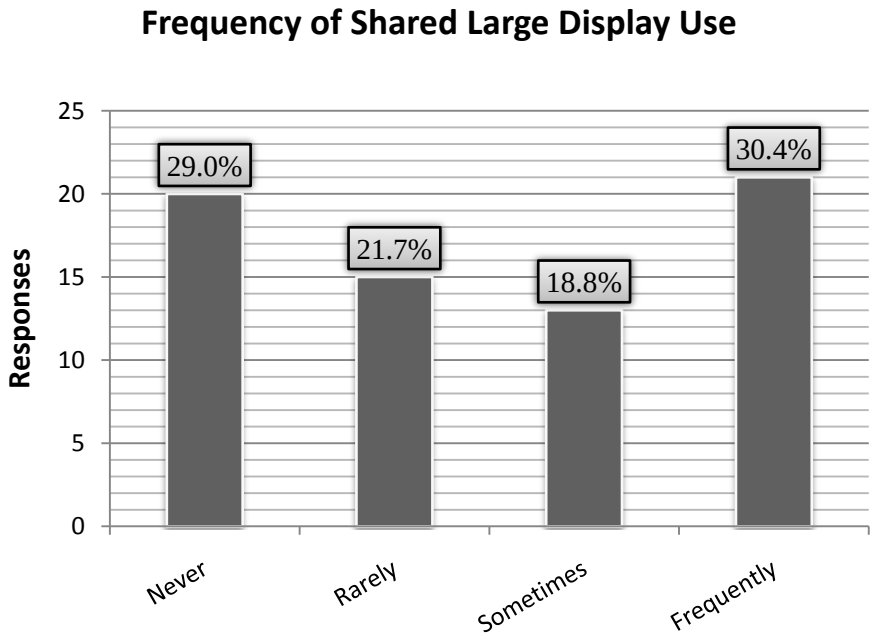


Figure 4.3: A breakdown of the frequency developers use a shared large display in their collaborative activities.

could be presented on a large shared display, we feel that it is fairly comprehensive within the domain of group-based development activities.

As can be seen from Figure 4.4, developers shared a large variety of information (and the applications that host/present that information) on the shared displays. Information particular to their work domain included source code, bug reports, design documents and diagrams. Additionally, the developers also shared email messages, screen shots, and websites. This result indicates that developers not only found value in displaying information from a large variety of applications onto the shared display, but also that they shared information from many different types of applications.

As we will discuss in the next section (4.2 Interviews), users preferred sharing information on the large display as it provided a shared artifact the group could take turns manipulating as well as providing them a signal visual focus that helped keep the collaboration better synchronized.

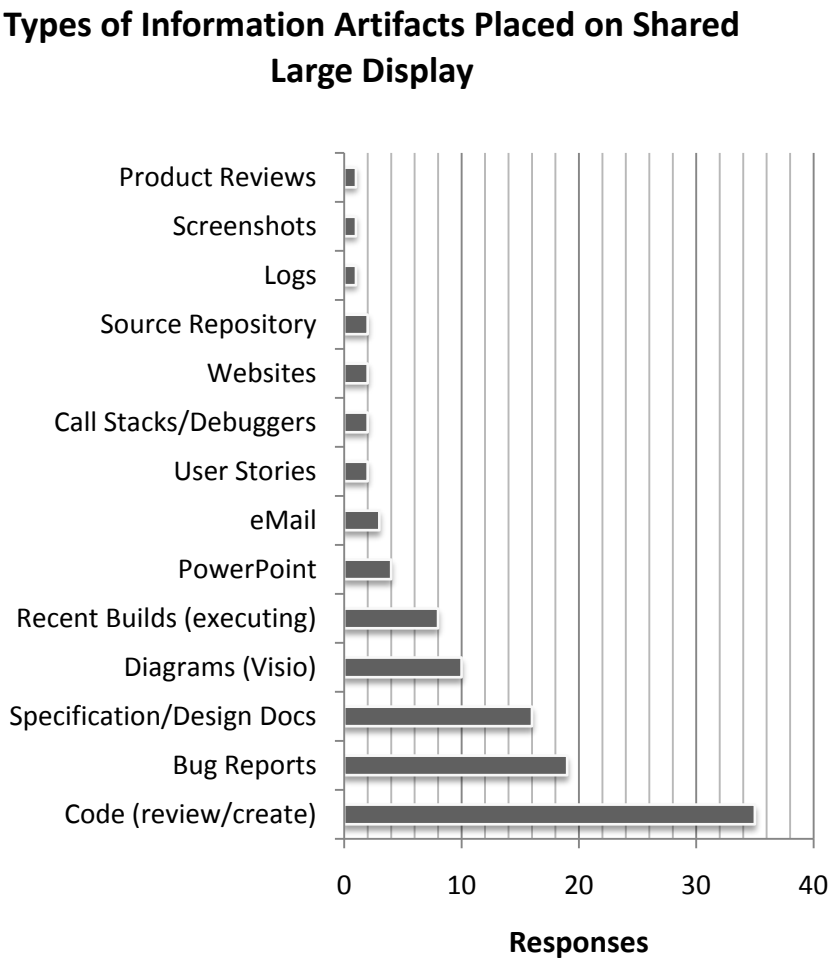


Figure 4.4: A breakdown of the different types of information artifacts that developers indicated are shown on shared large displays during collaborative activities.

The developers also provided details about the limitations of their current shared display configurations. Many users commented that their existing configuration only allowed one person to place and control content on the shared display. As one user states, it forces one person "to act as the presenter and that person drives the session or meeting." Another said they have to "share the display and take turns" projecting on the display. Users also expressed that the setup time required to place information on the shared display often made it "not worth the effort" to use for short collaborations.

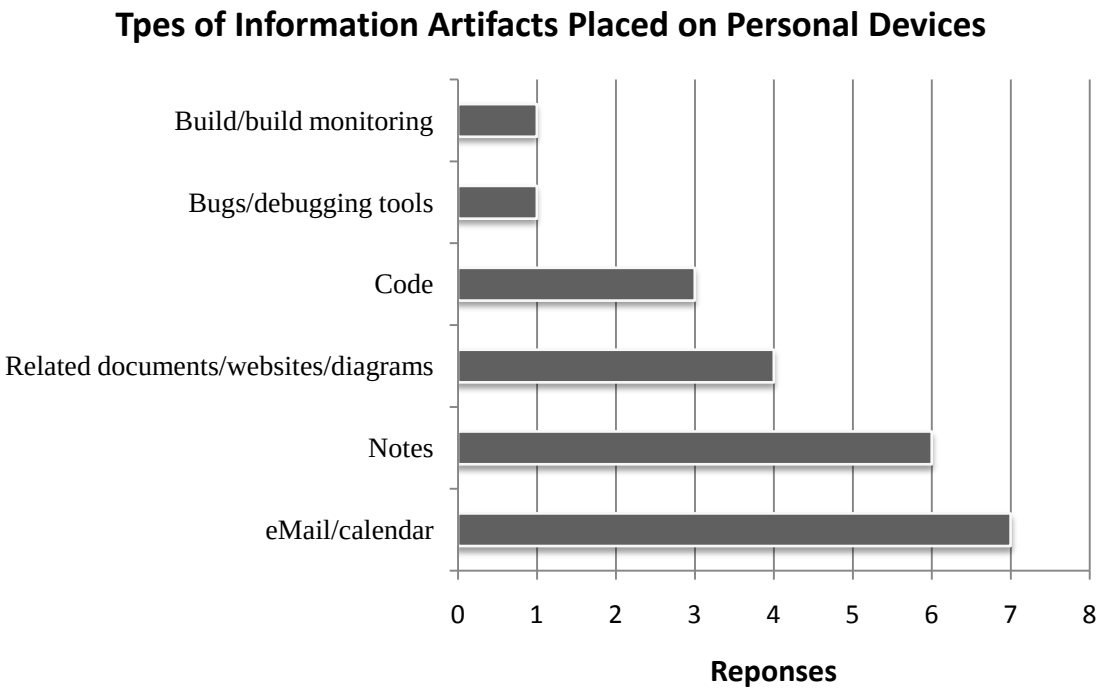


Figure 4.5: A breakdown of the different types of information artifacts that users indicated are placed on personal devices (e.g. laptops or tablet PCs) during their activities.

Overall, responses indicated that developers find great value in using the shared display to share myriad information artifacts to support their collaboration. However, usability limitations often limit the overall potential of the shared display to improve collaboration.

4.2.2.2 Use of Personal Devices

Our survey also asked developers to indicate what types of information artifacts are placed on personal devices during collaboration. In the survey, we defined personal device as a laptop or tablet PC that was not attached (output redirected) to a shared large display. Thus, any information placed on these devices was intended for use only by its owner. Figure 4.5 shows a summary of the various types of information artifacts that developers indicated they place on their personal devices during collaboration.

Like the shared displays, we found a large variety of information and applications being used. We also found through user comments that there was a desire to share this information across multiple personal devices (similar to sharing information on the shared displays). For example, the developers thought it would be useful for them to be able to collaboratively add and update shared notes. In the interviews we sought to further clarify exactly how personal devices, and the applications running on them, are used in the support collaborative activities.

4.3 Interviews

While the surveys provided insights into *how* users in our target population use MDE-like spaces to support their collaborative work, they did not provide details about users' *motivation* for the use of these spaces. We conducted 13 follow-up interviews to better understand the context of use for these spaces.

In our interviews developers were asked to describe the collaborative activities that they recently participated in. In particular, we asked them to describe the goal for the collaboration, what types of activities were performed, who participated in the collaboration, and what types of information was shared or exchanged and how it was shared or exchanged. The format of the interview was loosely structured; the interviewee was encouraged to be as detailed as possible in his or her description of group activities. The researcher would provide follow-up questions to interviewees to elicit clarification and, only rarely, would prompt the developer with questions to re-focus discussions on the interview topics.

4.3.1 Motivation for Collaboration

Developers described their work as being very complex software engineering projects, affixed to tight deadlines to deliver reliable products to market. One of many strategies for managing a project's complexity is to divide project requirements across team members. Similar to the findings in [37, 59, 80, 82, 129], we found developers would either be assigned a particular sub-set of the project's requirements (e.g. a specific functional unit or widget), or be assigned a specific role to perform on the team (e.g.

software tester). While this division of labor allowed for developers to have individual work, the nature of their projects (and the division of requirements) would often result in the developer being heavily dependent on other team members' assignments. This high degree of co-dependence required frequent collaborations to keep team members coordinated.

These types of coordinating collaborations would occur in both planned and ad-hoc situations. Many of the developers indicated their teams would conduct planned weekly or daily meetings where each developer would inform the group of his or her progress on their responsibilities and any difficulties currently being encountered. During these meetings, information was often externalized on whiteboards or shared large displays to better explain or demonstrate concepts. For example, to communicate progress with a particular requirement, a developer might execute a current build of the software, show code segments where bugs are occurring, or sketch a flow chart of part of the project's architecture.

Developers also indicated that collaborations between two or more developers would often occur on an ad-hoc basis, when a specific problem or goal needed to be addressed. Such collaborations were often not scheduled, or scheduled with only a few minutes or hours notice. The collaborations were problem focused and most commonly centered on bridging the developers individual requirements together, or would be cases where a senior, more experienced developer would assist a less experienced developer with a complex task. Similar to the team-wide collaborations, developers would collaborate in workspaces that contained whiteboards as well as personal devices and shared large displays (e.g. spaces that could be easily used as an MDE).

The motivation and style of the collaborations that we found in these interviews largely support the outcomes that were found from the surveys. A successful tool for enabling collaboration across multiple displays must provide an efficient mechanism for quickly sharing a diverse set of information artifacts while also allowing users flexible and low-overhead mechanisms for sharing control and input.

4.3.2 Workspace Configurations

Though not salient in our survey data, our interviews found two divergent strategies employed for team workspace configuration. The most common strategy was the traditional approach of giving each worker his or her own private office. Workers in this configuration spent most of their time performing individual activities within their offices. Participation in formal collaborations occurred through workers physically relocating to a shared meeting space such as a conference room or commons area; which were often equipped with a whiteboard and large shared display. Informal collaborations most often occur in or immediately outside personal offices. In these cases, workers share information by huddling around the office owner's desktop computer, or using a whiteboard.

In contrast to these traditional configurations, an emerging trend was for all of the developers on a team to work in a single, shared workspace. Often referred to as the "war room" or "bull pen," these spaces typically contain semi-private desks for each developer, multiple whiteboards or other vertical writing surfaces (e.g. opaque glass), and large, shared displays (e.g. plasma or projected displays). An example of one of these workspaces is shown in Figure 4.6. In these spaces, there was not as clear of a division between individual work, informal collaboration, and formal collaboration. While individuals still have individual assignments, developers said they would typically collaborate more often than their traditional counter part because the burden for collaboration was reduced by being co-located. For example, developers can just announce a question to the group rather than composing an email, sending an instant message, or walk to another person's office.

4.3.3 Use of Personal and Shared Displays

When asked about the frequency and utility of shared displays, the developers indicated the largest hindrance to their use is their availability. Specifically, many of the ad-hoc meeting spaces where collaborations took place did not have a shared large display present. Developers did however indicate that large displays are becoming more prevalent in these spaces. This feedback was consistent with survey responses and

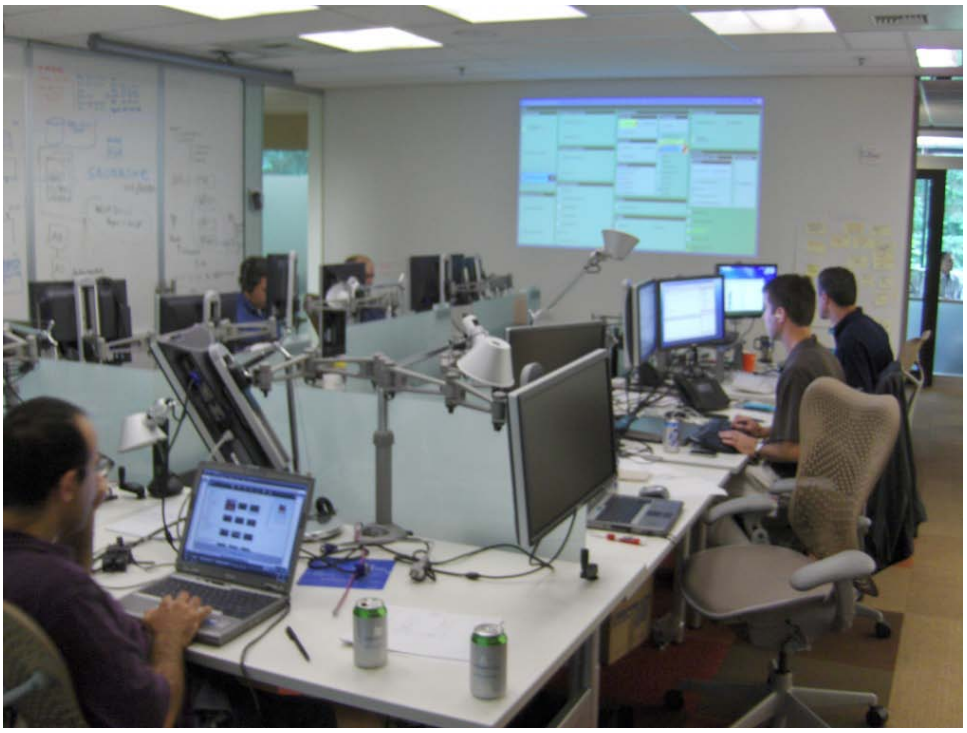


Figure 4.6: An example workspace of a co-located team. Each user has his own desk equipped with several monitors for working on individual tasks. A large, projected display is on the back wall. A cable to drive the display can reach each of the user's desks to enable them to hook up their personal devices to the large display to share information.

motivates the need for tools and systems that enable users to leverage these displays in support of their collaborative activities.

When shared large displays were present in the workspace, the users indicated the most significant hindrance was the process of connecting and configuring a shared display. Users said it was too time-consuming and distracted the group from the main collaborative task. Users indicated that a significant proportion of the information only needs to be shared "momentarily" and that it "isn't worth the hassle" for a short-lived gain. Finally, users said that when a large display is in use, it is usually configured such that one person's laptop is driving the display which usually restricts interaction with the information to be limited to only the laptop's owner.

Yet, with all the disadvantages of large shared displays, many users indicated significant value in using the shared displays to aid in their collaborative tasks. For example, many stated that the shared displays were very useful for performing group (or sub-group) reviews of code or demos of the latest software build. Further, users said they would be inclined to utilize shared large displays more often, if they were more available and the overhead for use was lower. Thus, a significant design consideration is to allow users to quickly place and easily interact with information on the large display.

Personal devices also played a key role in supporting users' collaborative activities. As discussed above, one primary purpose of bringing a personal device is to drive a large display. In addition, personal device are also commonly used to perform individual activities in parallel to the collaborative activity. For example, supporting the ability for users to take notes during a group discussion, use online resources to find information to assist the collaboration, and maintain awareness and access to personal communication tools (e.g. email and IM). Supporting results in [137], users believe access to their personal devices was essential to supporting their collaborative practices.

Despite their value, users also discussed many significant limitations of using personal devices to support collaborative activities. As noted before, users find it difficult to quickly place information from their devices onto a shared display. Likewise, users find it difficult and time consuming to share information between devices. For example, a simple task such as taking shared notes is still difficult to perform using existing applications and systems. Thus, tools and systems need to enable flexible models of information sharing, supporting the ability to easily exchange information across both personal devices and shared displays.

4.4 Requirements

Using the results of our surveys and interviews, we developed a set of core requirements that MDE frameworks need to satisfy in order to support effective collaborative work.

R1. *Serendipitous, ad-hoc collaborations require lightweight, low overhead interactions*. A significant burden of existing solutions is the overhead required to

configure the workspace, hardware, and software. An effective MDE framework needs to enable users to quickly link devices together and to facilitate sharing of applications "in the heat of the collaboration." That is, the framework's interface and interaction techniques must not distract users from their ongoing collaboration.

R2. *Any solution for sharing information must work with the tools that users already use*. A wide variety of complex tools were used by our study participants to support their collaborative activities. Such applications include code editors, debuggers, source repositories, bug databases, etc. These are applications for which users are already familiar and have very deep convictions about their use. Participants in our study were adamant that they would not adopt any solution that required them to abandon their current tools.

R3. *Sharing of information must be negotiated between artifact provider and consumer*. Users recognize many potential benefits of collaborating within co-located workspaces, but this does not mean that they want others to be able to push application windows onto their device at any time. There must be some form of negotiation, e.g., a user can signal when she needs assistance or has information that is needed by others; while other users can determine when to suspend their current task to offer assistance or choose when to view the information.

R4. *Multiple users should be able to simultaneously place information on a shared display*. For example, two software developers trying to integrate independent pieces of code could use the shared display to view both developers' code side-by-side, allowing for easily review, reflection, and modification of the code. Current techniques, such as emailing or sharing relevant files and opening on the device connected to the shared display, are too time consuming, disrupt the collaborative flow, and can require users to forfeit or transfer ownership of artifacts.

R5. *Multiple modes of collaboration should be supported*. For example, we found that users would work individually in parallel (e.g., each working on their own code), in subgroups (e.g., to correct a complex bug), or collectively (e.g., to review design decisions or assess project status). In addition, transitioning between these work modes was frequent and spontaneous; driven by demands of the ongoing programming activity. Thus, any new tools created for these types of workspaces must not preclude or inhibit any of these work modes.

What is important to point out is that while many existing framework provides support for some of these requirements, there is no framework that supports *all* of them. Further, while these requirements were derived from investigating how users in one particular task domain collaborate, there is nothing specific about the requirements that exclude them from being applicable to the general design of collaborative MDEs framework. For example, users in a variety of task domains perform ad-hoc collaborations, used specialized applications or tools, and employ a variety of collaborative methodologies.

Because of these limitations, we determined that a new framework would need to be built from the ground up. We discuss this new framework in Chapter 6. However, in the next chapter, we discuss lessons learned from applying the WIM interface to our target domain and the subsequent revisions that were made in an effort to create an interface and underlying framework design that best supported users' collaborative needs.

Chapter 5

Interface Revisions for the Selected Task Domain

In this chapter, we discuss our design revisions and evaluations towards applying the WIM metaphor to support users in our target domain. The process began with a series of paper-based low fidelity prototypes that were created to better understand how all the design requirements discussed in Chapter 4 could be adequately realized in a framework and supporting interface. While these prototypes are of user interface components, they also represent underlying framework functionality and capabilities. For example, UI controls for specifying permissions on shared applications affects and is influenced by how the underlying framework organizes and controls permissions. Thus, by performing evaluation walkthroughs of these prototypes with users, we are able to gain a set of user-focused lessons regarding the design of not only the user interface components but also the underlying functionality of the framework.

In the evaluation walkthroughs, we had exiting and potential users of MDEs assess the usability of prototype designs, the ability of the prototypes to support desired collaborative practices, and aspects of the designs they would change to better support those practices. Design lessons derived from this process were used to guide the implementation of our IMPROMPTU framework (see Chapter 6).

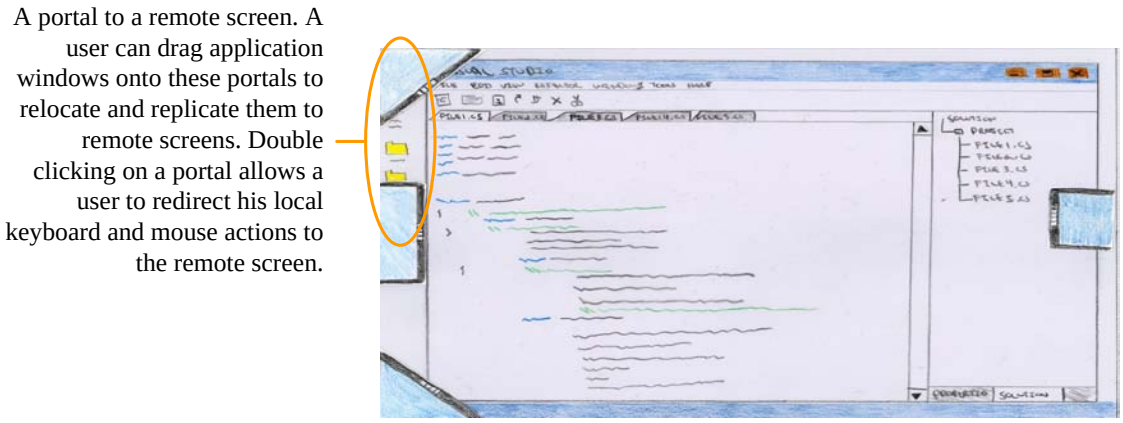


Figure 5.1: A scan of a paper prototype to support *quick action* mode. The prototype remote screen portals that appear on the periphery of a user's screen. The location of the portals on the periphery of the screen is relative to the location of the remote screen within the physical workspace. For example the portal in the top left is located forward and to the left relative to the user in the physical workspace.

5.1 Prototype Designs

The design process started with a focus on the WIM metaphor presented in Chapter 3. However, we quickly found that accommodating the revisions discussed in Section 4 of Chapter 3 along with the domain requirements presented in Chapter 4, Section 4 would create an overly complex and awkward design. For example, it was not clear how to build in the negotiated model for establishing replications of shared applications. As such, we began our design process by taking a slight step back, exploring different design and functionality features that, while not specifically WIM designs, did contain many core elements of the WIM metaphor.

Single-clicking on a portal brings up a full portal view of the remote screen. In this mode, a user can see thumbnail representations of the application windows running on the remote device. Within the portal view, a user can rearrange windows or drag them to another portal to relocate the window to another remote screen.

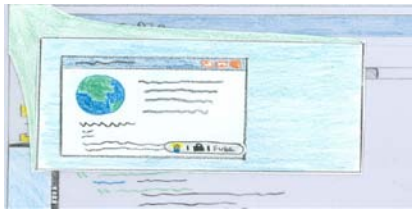


Figure 5.2: A close-up view of the *quick view* mode prototype that is showing the portal view for the top left remote screen.

One design, called *screen portals* (see Figure 5.1), was designed to provide users persistent access to shared displays, enabling users to quickly replicate application's output on these displays to facilitate quick sharing of information to peers.

To place content on a shared display using the *screen portals*, a user grabs the title bar of the application he desires to replicate and drag it to the edge of the screen. As the user drags the application, portals to available shared screens appear on the user's local screen. The portals appear based upon how the user is physically situated within the workspace. For example, the portal on the left side of the user's display corresponds to the display that is located to the left of the user within the physical workspace. Dragging and releasing an application window onto a portal initiates the replication action (creating a shadowed copy on the remote screen while still keeping the application window open on the local screen).

Performing a single click of the screen brings up a full portal view of the remote screen (Figure 5.2), allowing the users to select the application window, drag it out of the portal view and releasing it in his local desktop. The portal view can also be kept open to provide the user with an ongoing awareness of the information items being presented on the remote screens.

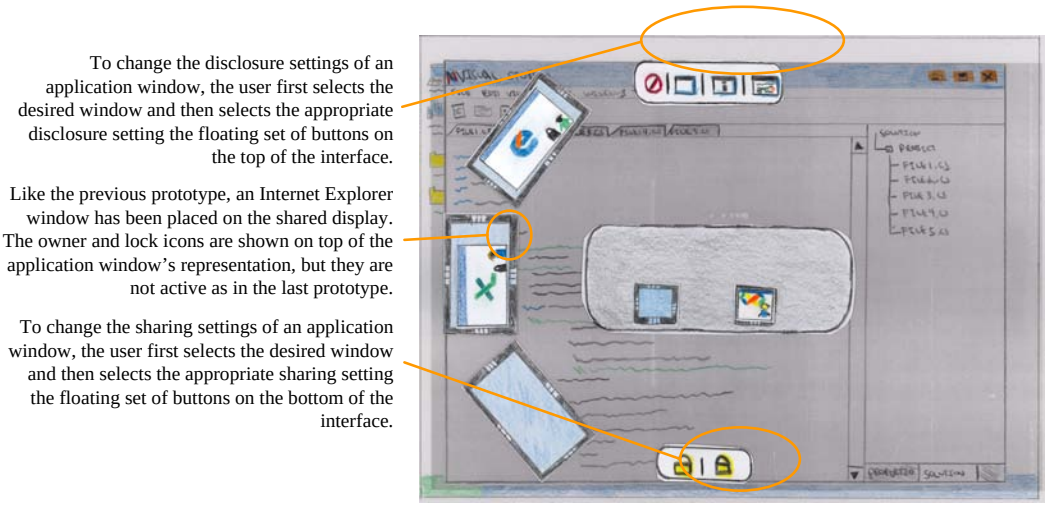


Figure 5.3: A scan of the paper prototype that allows users to control how applications are shared and represented. The call-outs show the centralized controls for specifying sharing and representation preferences.

The *screen portals* also enable a user to redirect his local input to remote screen. By hovering his mouse cursor for short time near the edge of his screen, the portals will appear allowing the user to double click on a shared screen to redirect input to that screen. To return input to the local screen, the user can use the same interaction with the portals on the shared screen, or use a hot key sequence.

A second design, more closely models the world-in-miniature (WIM) metaphor discussed in Chapter 3, was developed to explore methods for controlling the availability of shared applications. As illustrated in Figure 5.3, applications are represented on the device in which they are currently being displayed. For example, the device on the right side of the table has an instance of Visual Studio running, as indicated by the Visual Studio icon. In addition to the application icon, two addition visual indicators are shown on each application; a user icon representing which user is the owner of the application, and a lock icon representing the current sharing mode the application is in. Applications that a user owns (i.e. the application was started on and relocated/replicated from the user's local device) can be transitioned between *show mode* (other collaborators can only view the application) and *share mode* (all collaborators can interact with the application).

Within in this prototype, the lock icon of owned applications can be toggled; locked to represent *show mode* and unlocked to represent *share mode*.

In addition to the lock icon, a user can also adjust how applications in *show mode* are represented in other users' interfaces. User can select from full disclosure (full thumbnail when zoomed), icon only disclosure (application detail limited its icon), generic disclosure (only an outline of the application is shown, with no icon to associate type of possible content), and invisible (application not represented to other users). A user can toggle between *show* and *shared modes* by first selecting the application they wish to change, and then pressing the lock or unlock button icons. A similar interaction is used to change an applications level of disclosure.

Both prototypes are designed with the assumption that applications would by default be set to the invisible level of disclosure. When a user transitions an application to a different level, the application's sharing modality would be set to *show mode*. These defaults would allow users to balance being able to keep their fellow collaborators aware of their current activities, while also being able to protect their privacy for certain types of applications.

A third prototype explored the ability for users to manage and organize shared applications based upon the tasks or sub-tasks those applications were supporting. Similar to facilities provided by GroupBar [130] and Scalable Fabric [111], users are provided a task tool box that organizes shared applications by task (see Figure 5.4). Users drag an application's representation onto a particular task to categorize the application to that specific task. This design has the particular advantage in that it could be kept open on the user's screen independent of the icon map, allowing for quick awareness of what applications are being added to the workspace and how they are being assigned.

5.2 Evaluation Methodology and Procedure

Keeping within our target domain, we evaluated the prototypes with 7 software developers from 2 software companies. 4 of the participants in the evaluation were

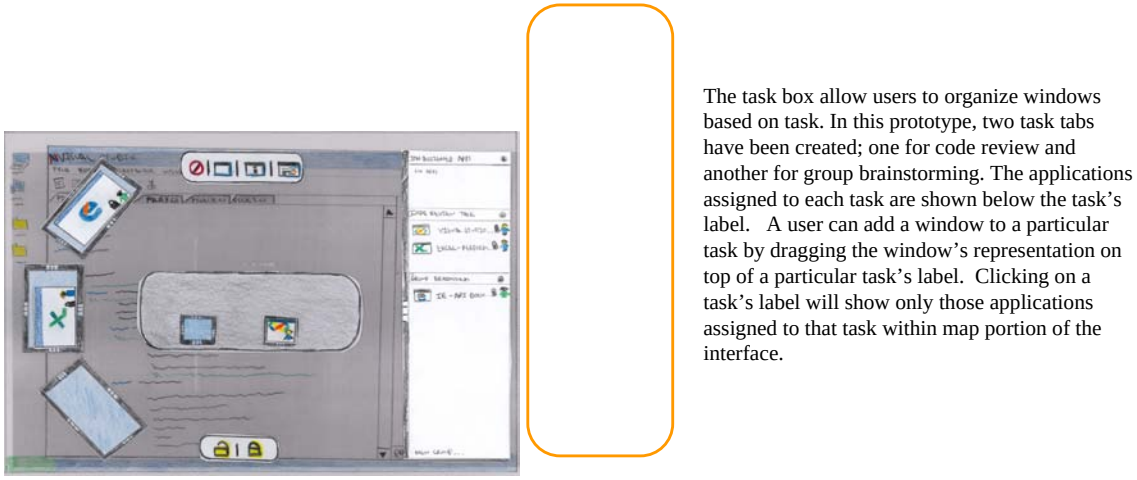


Figure 5.4: A scan of the tool box based prototype that provides users the ability to organize applications based on task or sub-task.

programmers in a game development studio. These participants were a part of a large, ongoing development project (over 50 developers). The remaining 3 participants worked on performance profiling tools in a division of a major computer hardware company. These participants worked in a relatively smaller team (~12 developers). Evaluations were conducted individually, each lasting (on average) about 1 hour. 5 evaluations were conducted at the participant's place of work. The remaining 2 evaluations occurred in our research lab.

Evaluations were split into two parts. The first part was a structured interview to gather details about the participant's development responsibilities, current team practices and methodologies, collaboration styles, and overall dynamics of his or her work environment. This process provided a baseline understanding the participant's typical collaborative practices which could then be compared to the practices of the users that participated in our surveys and interviews. Understanding such differences is important as they could affect how certain features of the prototype were used and/or appreciated.

Next, the researcher would explain the concept of an MDE and provide example scenarios of how MDEs could be used to perform tasks. The prototype's features were

then described and demonstrated to the participants. Using the information from the structured interviews, the researchers would then ask the participant to demonstrate how he or she would use the prototype to complete commonly performed activity. For the particular domain studied, such tasks included code reviews, paired programming, and debug sessions. Through this process we were better able to understand the strengths and weaknesses of each of the prototypes and ultimately uncover more lessons from improving the design of MDE frameworks.

5.3 Lessons and System Requirements

The evaluations highlighted many strengths and weaknesses of each prototype. These lessons are summarized below.

L1. *Users favor a more people-centric, rather than device-centric representation of personal devices*. When using the prototypes to complete tasks, we found that participants would first identify the owner of the relevant information artifacts and then map the owner to a device within the workspace. For example, when explaining how he would use the prototype to perform a code review, one participant stated that he would first identify the developer whose code is going to be reviewed, identify that user's device within the interface, and then identity the appropriate applications to be replicated onto a shared display for group review. To avoid the person-to-device translation, participants felt the interface should simply represent the users participating in the collaborative session and provide direct access to their shared applications.

L2. *Users appreciated an always-in-view visual representation of the workspace*. Although users found the task-based grouping too cumbersome, they did appreciate how that prototype allowed a list of participants and their shared applications to be in view at all times. Users felt this compact view would allow them to easily maintain an ongoing awareness of peers' activities which was useful for identifying opportunities for initiating collaborations. For example, one user stated that he could monitor his manager to know when he was working on

the project's timeline. In doing so, this user could bring forward relevant issues when it was within the working context of his manager.

L3. *Only a few levels of sharing and disclosure are necessary*. Users overwhelmingly indicated that the prototypes had far too many different levels of sharing and disclosure than was necessary. The consensus of users was that if the application is available to the group, then the group should be able to see all the details of that application (i.e. partial disclosure was not useful). Users did however feel it was necessary to control shared input and appreciated the lock/unlock metaphor. Thus, future frameworks should only consider implementing three modalities: private (no group access), show (view only), and share (joint access).

L4. *Portal views were preferred for shared screen interaction*. Users found a major benefit of the portals interface to be its ability to facilitate making information quickly viewable to the group. For example, one user said that he could foresee using the portal interface to quickly "throw" code on the projected display to get quick feedback from his peers.

L5. *While valued, users felt the spatial representation of devices would be too burdensome to maintain*. Users indicated that they would often arrange themselves and their devices based upon the tasks and how they decided to use their devices to support those tasks. For example, one user said if they were doing bug reviews, they would arrange themselves and their devices so that they could see both the shared screen and their personal devices at the same time to facilitate comparing information across screens. In another example where a user described a brainstorming session, he said they would often arrange themselves to facilitate better "eye-contact." Without an automated approach to updating the spatial representation, users believed the spatial interface would often become non-representative of the actual workspace.

These lessons highlight the additional insights provided by our in-depth investigation into the use of an MDE to support collaborative activities within a specific task domain. For

instance, L5 reinforces lessons from Chapter 3 that show the value of a spatial representation. However, it also provides insights that a spatial representation may not be practical in all cases. Combined with lessons from previous chapters, these lessons contribute to developing a further understanding of how to design interfaces and their supporting frameworks to support co-located collaboration in MDEs.

In the next chapter we present IMPROMPTU, a new framework from supporting collaboration within MDEs. IMPROMPTU is the culmination of a truly multifaceted design process; building first on core groupware requirements (Chapter 2), leveraging lessons derived from the design and evaluation of previous MDE interfaces (Chapter 3), and incorporating domain specific design context from our target domain of group-based software development (Chapters 4 and 5). After we introduce the interface and describe how it is used, we explicitly discuss how design elements from our previous WIM interfaces were incorporated or adapted into the design of IMPROMPTU.

Chapter 6

IMPROMPTU: A New Framework for Supporting Collaboration in Multiple Display Environments

In this chapter we present IMPROMPTU, a new interaction framework for MDEs. A key innovation of IMPROMPTU is the ability for users to easily share any off-the-shelf application across any number of devices. Such devices could be collaborators' personal devices (e.g. laptop or tablet), a shared display (e.g. plasma display), or a combination of the two. In addition, IMPROMPTU provides a novel, people-centric user interface that allows users to set which local applications are available to the group, specify how others in the group can use those shared applications, and instantiate replications of applications that others have shared. More broadly, the functionality provided by IMPROMPTU enables users to better leverage the collaborative benefits of an MDE to support their group activities.

The IMPROMPTU interaction framework is comprised of three main components:

- *User Interface*. The user interface provides the interaction mechanism for users to leverage the underlying framework functionalities. Specifically, the user interface allows users to identify and maintain awareness of peers participating in the collaboration session, specify and manage which of their local applications are available for use by peers, instantiate local replications of applications made available by peers, and organize shared applications on large displays present within the MDE.

- *Replication Services*. The replication services provide support for application sharing; which includes application window content streaming to replicating peers, coordination of multi-user input, and controlling availability and level of interaction.

- *Coordination Server*. The coordination server centrally manages the state of the MDE across all participating devices and users. This includes insuring all running instances of the user interface are up-to-date and boot strapping replication instances.

6.1 User Interface

The user interface provides a visual representation of the group members and large displays, and the application windows that have been made available to the group and those that have been placed on the large displays. As shown in Figure 6.1, the interface is comprised of three main components: Collaboration Control, Collaborator Bar, and Shared Screen Dock(s).

6.1.1 Collaboration Control

This control allows a user to configure whether an application window is available to other group members, and if they are allowed to modify or only view the content of the window. See Figure 6.2. The control is automatically displayed on the title bar of every top-level application window. This location was chosen to reinforce the concept that this is a window-level operation, provide quick access to the functionality, and provide a persistent indicator of the window's sharing state.

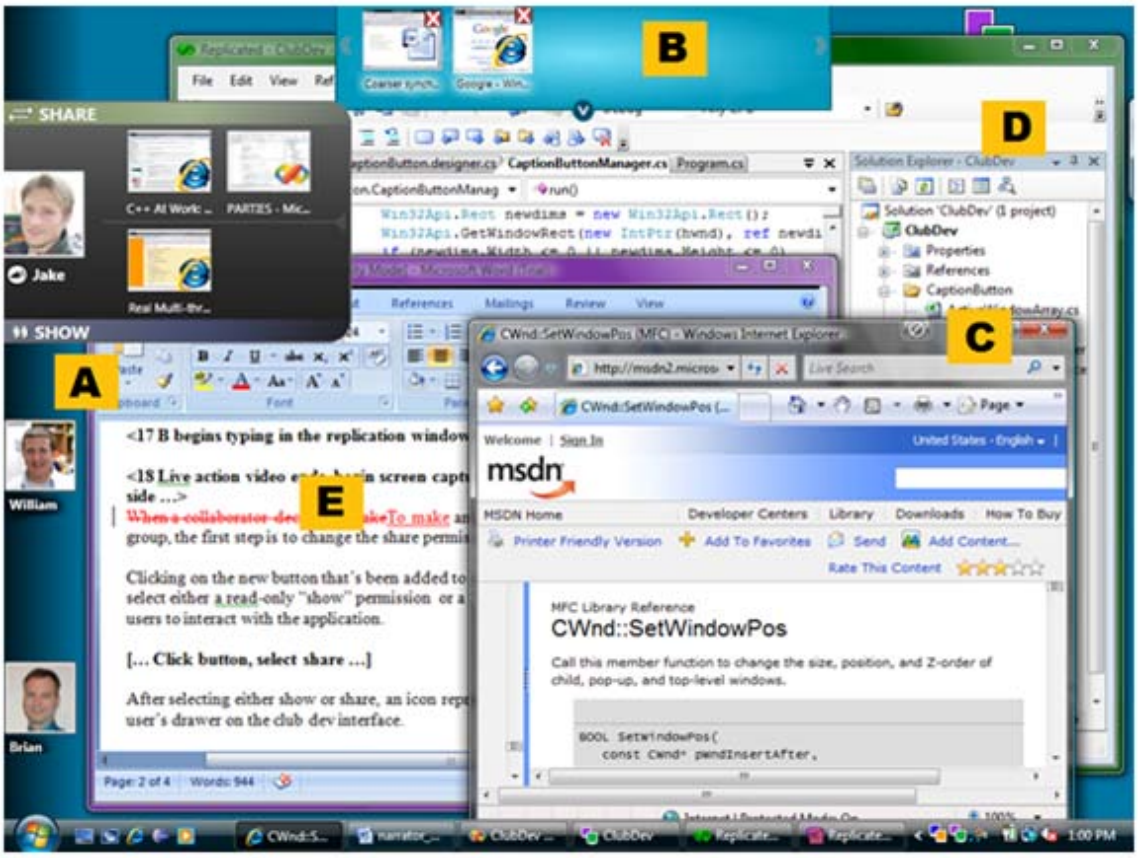


Figure 6.1: Screenshot of the IMPROMPTU user interface, along with replicated and local application windows on a user's personal device. The collaborator bar is on the left (A), and one collaborator drawer is expanded showing the applications available to the group. The shared screen dock (B) allows windows to be placed on a large shared display. Whether an application is available to the group and what level of control is allowed can be set using (C). A replicated window in share mode allows interaction with its content (D); while a replicated window in show mode allows a user to view, but not modify its content (E).

Selecting the control reveals three sharing options (Figure 6.2):

Do not show or share. The window remains private and is not available to group members, and this is the default value.

Show. The window is available to the group, but in a view-only mode. A live thumbnail of the window is displayed within the *show* area of the user's representation in each group member's Collaborator Bar.

Share. The window is available to the group and anyone can modify its content. A thumbnail of the window is displayed within the *share* area of the user's representation in each group members' Collaborator Bar.

Offering both *show* and *share* is necessary as we found that users typically have a strong sense of ownership over source code and related artifacts (See Chapter 4, Section 4.3.3). For example, a user can set a window to *show* to allow others to maintain awareness of her activity in relation to that window, but not be able to interact with it. Whereas with *share*, group members could edit a source code file or other document together, or a user could pass control temporarily to another group member.

6.1.2 Collaborator Bar

The collaborator bar provides a representation of each user participating in the collaborative session and the application windows that each user has made available to the group. When a user joins a session, her personal photo (or other selected image) appears within the Collaborator Bar, located on either side of the screen. See Figure 6.1 (A).

Each user's representation in the Collaborator Bar has a drawer with two rows. The top row summarizes the application windows that have been set to *share* while the bottom summarizes the application windows that have been set to *show*. See Figure 6.3 (A). Clicking on the expand arrow on the right side of the drawer provides an expanded view of the applications and allows users to directly select particular representations of applications. See Figure 6.3 (B).

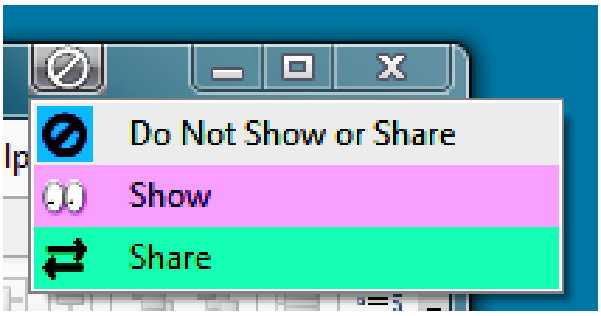


Figure 6.2: The collaboration control is displayed on every top-level application window. It is used to configure whether the window is available to the group, and whether group members can only view the application window (Show) or interact with it (Share).

From a group member's expanded view drawer, a user can drag the desired application representation and drop it onto the desktop, causing a replication of the source window to be rendered. For example, in Figure 6.1, a user has created a replicated view of a team member's Visual Studio window. If the owner sets the window to *share*, the user could edit the code while the owner switches to another task, or the two could edit the content near simultaneously. To help establish presence, provide awareness, and improve coordination, IMPROMPTU renders tele-pointers whenever the owning or replicating users' cursor focus is within the shared application. When a user does not have a shared window in focus, the tele-pointer is not rendered. The tele-pointer were implemented to be consistent with the past systems' use of the technique [17].

The interaction needed to replicate an application window embodies a desired negotiation process. For example, it is the owner of an application window who determines if it is available to the group, while it is each group member who decides if and when to create the replication of a window.

6.1.3 Shared Screen Dock

The shared screen dock allows a user to place a replication of an application window on a large (shared) display, organize the windows remotely, and redirect input to the display to

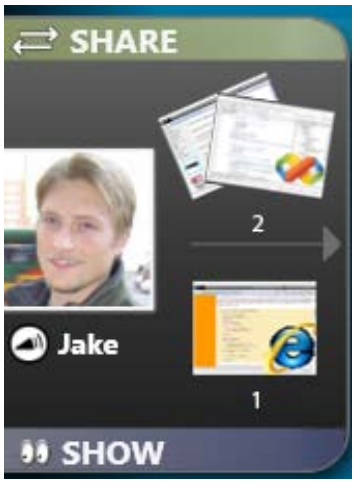
interact with replicated windows. Our system can support multiple large displays, and each display would be represented by its own dock. Any user can place any number of replicated windows onto the displays.

The dock is minimized by default, and opens when the user moves the cursor over it. When opened, the dock shows thumbnails of all the application windows on the corresponding large display. In this view, the thumbnails are shown left-to-right, as this allows all of them to be seen at once without occlusion (Figure 6.4 A).

When the expand button (bottom of an opened dock) is selected, it expands and displays a miniature representation of the windows on the large display (Figure 6.4 B). By interacting with the thumbnails in this view, any user can adjust the position and z-order of the remote windows.

Any application window that has been made available (set to *show* or *share*) can be placed on a large display. From any group member's representation in the Collaborator Bar (including her own), a user drags the representation of the desired window and drops it onto the appropriate screen dock. The dock expands and the user can position the replicated window as desired. A large display can contain replicated windows from multiple users *at the same time*. A replicated window is removed from a large display by selecting the close icon at the top right of its thumbnail.

In contrast to other interfaces for multi-device environments [22, 127, 147], our interface does not provide a strict spatial representation of the workspace or a portal view of applications on personal devices. As discussed in Chapter 4, participants in our initial studies expressed that being able to view how applications are arranged on a group member's personal device provides little value and they wanted the interface to emphasize the people they were working with rather than the relative location of their devices.

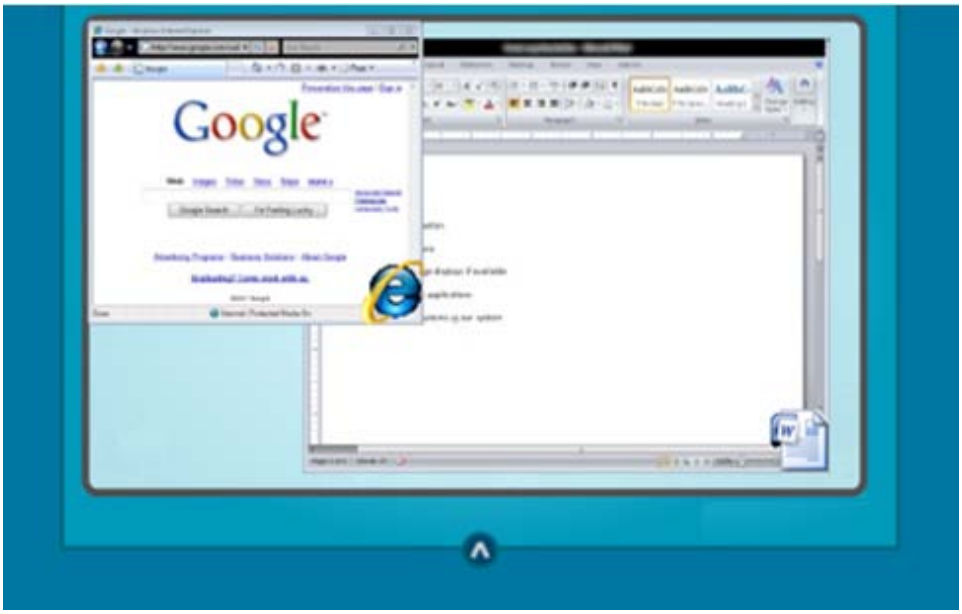


(A)

(B)

Figure 6.3: A close-up of the summary (A) and expanded (B) views of the collaborator drawer. This particular user has two application windows in share mode (Internet Explorer and Visual Studio) and one window in show mode (Internet Explorer).

(A)



(B)

Figure 6.4: The shared screen dock. Hovering over it gives a summary of windows placed on the corresponding shared display (A). Selecting the arrow causes it to further expand, providing a view that allows windows to be organized on the shared display (B).

6.1.4 From World-in-Miniature To IMPROMPTU

While IMPROMPTU's interface inherits many design elements that were a part of the World-in-Miniature interfaces, it is also different in many respects. The design shift between WIM and IMPROMPTU represents an evolutionary, iterative design processes that was followed as we worked to create an effective solution for a particular target task domain. As we describe in Chapters 4 and 5, our design focus shifted from supporting

basic management tasks to supporting the overall collaborative practices of the users within the domain of group software development. Thus, IMPROMPTU is the result of a design process that built on past lesson from the WIM interfaces *as well as* lessons that were learned from our in-depth investigation of collaborative practices within the domain of software development.

One large design shift in IMPROMPTU was moving from a device-centric representation of personal devices to a people-centric representation. As we discuss in Chapter 3, Section 4, users found it difficult to map device representations to their owners and/or role in the collaboration. For example, it was difficult to identify which device on the table belonged to a particular user. Additionally, we found for our particular target domain, roles and/or individual responsibilities in tasks are nearly always well defined. As a result, ownership of information artifacts is also strong and critical to the support of the group's collaborative activities (see discussion in Chapter 4 for further details). For example, we found when a user opened code in his or her code editor they were essentially taking possession and responsibility over the artifact. Thus, a common behavior of users for identifying shared artifacts was to first identify the artifact's owner and then browse the available artifacts on that individual's device.

With IMPROMPTU, we designed the Collaborator Bar to enable users to more easily identify peers participating in the collaboration and better facilitate exchange and control of shared artifacts. While the Collaborator Bar does not provide the full spatial layout of users (of their devices more specifically) it can be customized to incorporate some spatial layout characteristics. For example, the Bar can be repositioned onto various sides of the screen and users in the Bar can be repositioned to better match their physical location. While it's not possible in the current iteration, future designs could allow for a user to "undock" a collaborator and freely position that user on the local desktop. This behavior is similar to Microsoft's Vista SideBar widget undocking behavior [6].

In IMPROMPTU the spatial layout of shared large displays was also relaxed. In IMPROMPTU the large displays are, like the Collaborator Bar, allocated on the periphery of the user's screen. The design change was motivated by two main factors.

First, and most importantly, the software developers we interviewed and observed stated that an essential requirement of and MDE interface is the ability to have quick and efficient access to the shared displays. In our interviews, many developers described an interaction style that was similar to what was eventually implemented in IMPROMPTU's Shared Screen Dock. For example, several users described a "drawer" feature for placing information on a shared screen. Others described a "tunnel" or "vortex" where applications could be sent to the shared display. Similar to the Collaborator Bar, a future revision of the interface could allow for the screen docks to also be "undocked" allowing for users to form more discrete spatial arrangements of displays that are more akin to the WIM layout.

The Shared Screen Dock does, however, include many WIM design elements. For example, we included many of the affordances provided in SEAPort's zoomed-in interaction technique into the current interface. For example, when the dock is expanded it provides a live, thumbnail spatial layout of the shared applications currently being shared on the large display. This allows users to change the z-ordering and placement of shared applications.

Beyond shared screen control, there are many interaction features in IMPROMPTU's interface that are inherited from the WIM designs. For example, similar to the scalability improvements we included in the SEAPort interface (see Chapter 3, Section 3), IMPROMPTU's Collaborator Bar also provides icons and live thumbnails of applications shared on a personal devices. The application sharing control settings (Chapter 3, Section 4) are nearly identical to the Collaboration Control interface component of IMPROMPTU. More generally, IMPROMPTU also preserves a direct manipulation interaction technique that allows users to visually identify users, devices, and applications and supports users ability to manage information within an MDE using the many different input devices that may be present within the workspace (e.g. touch, stylus, and keyboard/mouse).

6.2 Replication Services

As we discussed in Chapter 4, a central requirement of our framework is to allow any off-the-shelf application to be shared and interacted with across devices. This goal is important because it would allow users to continue using the applications that they prefer and need for their daily work activities (See Chapter 4, Section 4.3.3 for further details).

Our approach is to use a *replication model*. In this model, the pixel data associated with an application window in the frame buffer is captured, and is then available to be sent (replicated) to other devices in response to user requests. Interaction with a replicated window is enabled by capturing input within the replicated window and sending it to the corresponding source. Advantages of our approach include:

- Any off-the-shelf application can be replicated across devices *and* its interaction context is maintained (e.g. views, highlights, undo stacks, and breakpoints).

- Only one device needs to have the application installed in order for it to be utilized by the entire group.

- The owner of the application window maintains control over its content. For example, the owner can choose to discontinue sharing the application window at any time.

- Input originating from replicated windows does not interrupt the input stream on the owner's device.

- Users can interact with a replicated window even when the owner of the application does not have it in focus (e.g. it is minimized or occluded from view).

These advantages are important for many forms of co-located collaborative work. This is particularly true in situations where users often utilize different tools, applications typically reflect a rich interaction context (e.g., breakpoints in a debug window), and

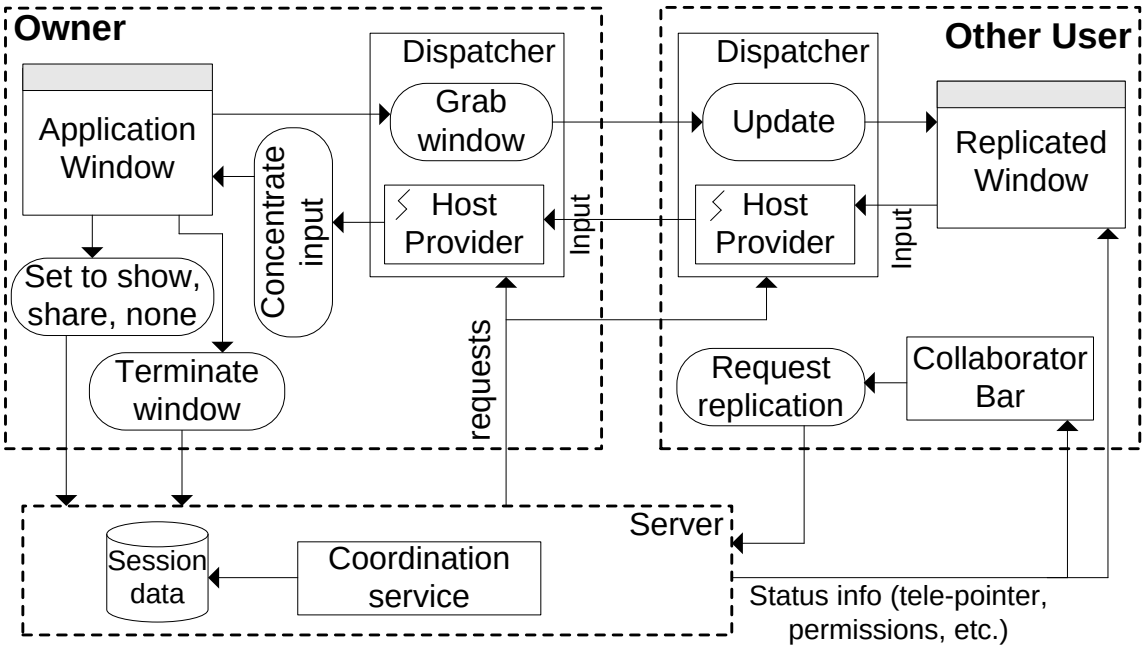


Figure 6.5: The system architecture of IMPROMPTU. Any user can replicate any application window that has been set to *show* or *share*, and interact with windows (and their content) that have been set to *share*.

users have a strong sense of ownership over content. As illustrated in Figure 6.5, the replication services are comprised of three main components: the dispatcher, the host provider, and the input concentrator.

6.2.1 Dispatcher

The dispatcher is responsible for establishing and maintaining replicated windows between devices. The dispatcher executes on every device participating in the session, and listens for replication initiation events from the Coordination Server. When an event is received, the dispatcher on the source device establishes a network connection with the dispatcher on the destination device. New threads are then spawned to coordinate the point-to-point stream of the window's pixel data and input between devices.

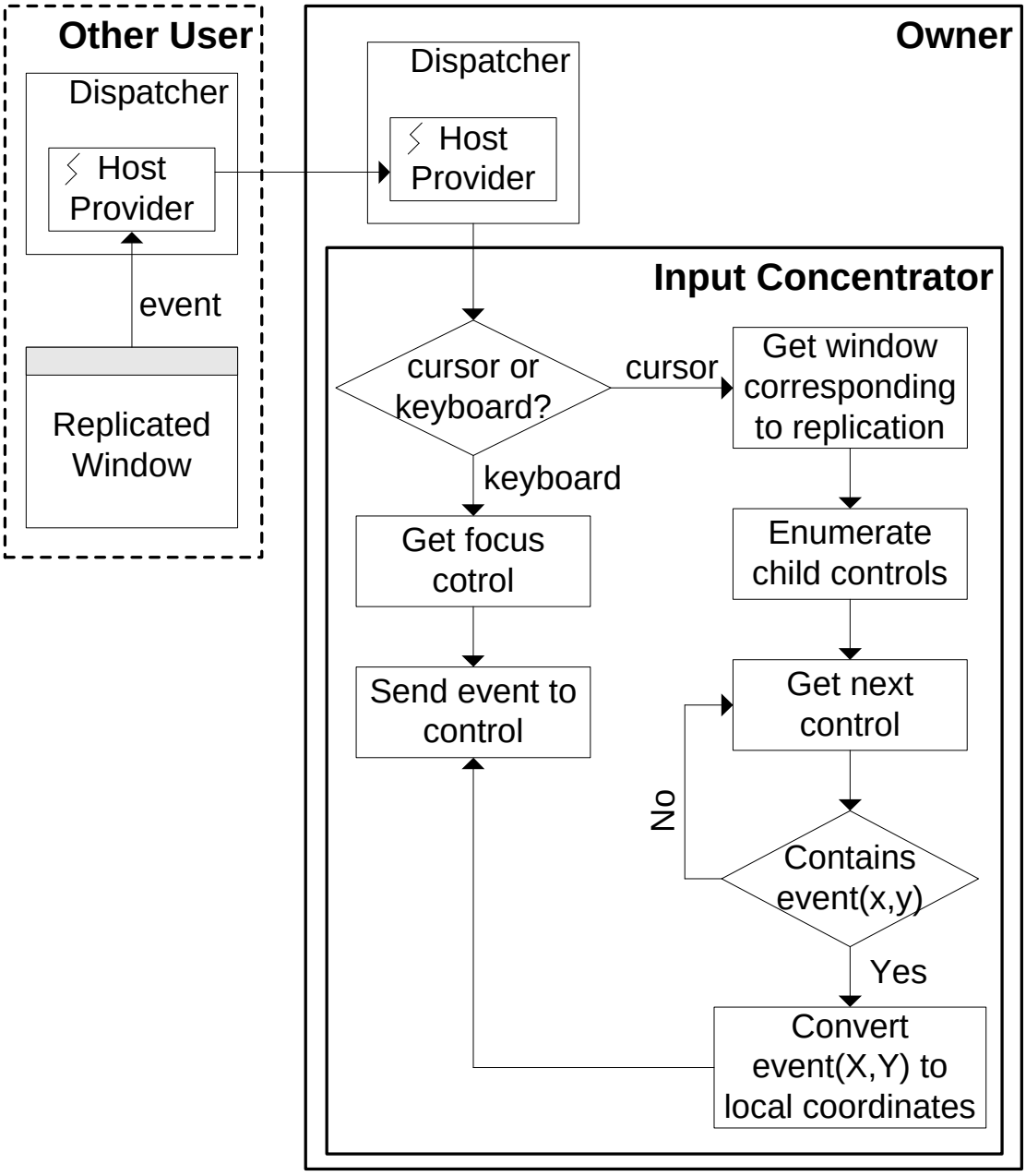


Figure 6.6: Control logic of the Input Concentrator. Other users are able to interact with replicated windows without interrupting the input stream of the owner. Users can interact with replications even if the source window is minimized or not in focus.

6.2.2 Host Provider

When an application window is replicated, an instance of the host provider is executed on both the source and destination device. At the source, the host provider is responsible for capturing and streaming the pixel data of the application window. Capturing frames of the application is performed using `PrintWindow` and `BitBlt` calls within the Windows API. These calls allow a window's pixel data to be captured even when the window is not in focus or at the top of the z-order.

A special case is when an application window is minimized, as it is no longer redrawn. To handle this case, the host provider overrides the minimize operation such that the window is actually moved into a non-viewable portion of the video buffer. This allows the window to continue to be available to remote users, yet is completely transparent to the owning user (they still perceive that the window has been minimized).

On the destination device, the host provider is responsible for rendering the replicated window. As illustrated in Figure 6.1, replicated windows are drawn with a different colored title bar and border to differentiate them from locally running applications. One color (green) is used for replicated windows that are in s*hare* mode, whereas another color (purple) is used for windows that are in s*how* mode.

6.2.3 Input Concentrator

A common technique for allowing multiple users to interact with an off-the-shelf application is to multiplex the hardware cursor [18, 27, 74]. A known limitation of this approach is that other users' interaction with the application causes temporary interruption to the local user's input stream. For example, two users would be unable to each simultaneously interact with different applications because they would always be competing for input focus.

We are leveraging a more advanced technique for enabling multi-input with existing applications, which addresses the above limitation. Our technique is to place incoming events not on the global event queue, but to send them directly to the containing UI control, as depicted in Figure 6.6.

As a user interacts with applications on the local machine, the host provider intercepts each event and checks if it corresponds to a replicated window. If not, it is ignored. If it does, the host provider removes the event from the queue and forwards it to the Concentrator on the owning device.

When a forwarded event is received, the Concentrator identifies the source window, enumerates its child controls, identifies the control that contains the event, transforms the coordinates of the event relative to the control, and 'sends' the event directly to the control. This technique has proven to be extremely flexible. For example, it allows other users to interact with a replicated window even when the source window is not in focus or has been minimized. It also allows two or more users to each be interacting with a different application hosted on the same device without conflict.

6.3 Coordination Server

The coordination server maintains the state of the collaborative session, including IP addresses of participating machines, sharing status of application windows, source/target pairs for replicated windows, etc.

The coordination server broadcasts updates to all participating devices whenever the state of the session changes (e.g., a user joins or leaves the session, or the sharing status of an application window has changed). The coordination server is also responsible for sending replication dispatchers' requests to instantiate a replication of an application window.

The coordination server is a publicly accessible service that manages all of the collaborative sessions. When IMPROMPTU is first launched, the user is presented with a configuration interface. The interface is used to initiate or join a collaborative session, specify whether the device is a personal device or large display, and configure personal settings such as the user's icon that appears in the collaborator bar.

6.4 Implementation

IMPROMPTU is fully functional and was written mostly in managed C#, though some of the lower-level components were written in unmanaged C++. The interface runtime is built using the Windows Presentation Foundation libraries that are part of the Microsoft .NET 3.0 framework. The replication services are built on top of the Win32 API 6.0.

For reliability and scalability, the coordination server is implemented as a set of tables, stored procedures, and notification services loaded on a Microsoft SQL Database Server. While our system currently works with Windows Vista, it is also compatible with legacy Windows-based operating system such as Windows XP. It could also be ported to other operating systems as the techniques that we have developed could be mapped onto other commonly used windowing systems.

6.5 Initial Usability Study

The overall goal of IMPROMPTU is to provide a fully implemented framework that is capable of being deployed within an authentic task domain that can evaluated to understand its impact on collaborative practices. Considering the relative high cost of preparing and performing a field study, we first conducted an initial usability study to identify potential usability and software stability issues, develop a basic understanding of how users leverage the framework to support collaborative tasks, and to test a backend logging tool for automatic collection of framework usage data. Using the results of this study, small but significant improvements were made in the usability and stability of the framework. Further, the context of use observed in this initial study was leveraged to develop a collaborative behavior observational coding scheme that was later used in the field study (See Chapter 6). The later field study was also performed with co-located software development teams.

6.5.1 Users and Task

We recruited 4 project groups of 2-3 individuals from a senior-level software engineering course in Computer Science Department at the University of Illinois. Since the course

spans multiple semesters, we were able to recruit groups that had been working together on a project for several months. Thus, these teams were already comfortable working together and had an established practice for subdividing and coordinating labor on collaborative tasks. Additionally, we felt these groups would be interested in trying out new tools for enabling more effective collaborative work.

Each team was asked to perform a basic programming task. The task was carefully designed to be sufficiently challenging for the group, but not so complex that meaningful progress could not be made in the time allotted. Also, the task was substantial enough to allow users to explore the use of the framework in the context of supporting the collaborative task.

Each team was asked to build a personal bookmark manager with the following functionality:

- Add bookmarks, which were composed of URLs, titles, user-defined comments and last access date.
- Remove bookmarks.
- Present users with a table of bookmark entries.
- Allow users to search comments (string matching).
- Allow users to open bookmarks in a web browser directly from the bookmark manager.

Though groups were informed that they could perform the task however they preferred (e.g., by working jointly), we recommended that the task be subdivided into functional components, and each person be assigned one of the components. For example, one person could create the component that parsed the input data, another could develop the

graphical interface, and third could handle the search. The group was asked to produce a single integrated demo.

Because IMPRMPTU had already been tested with MS Visual Studio, we instructed the group to use this particular application for generating the source code. However, the group could use any language supported by Visual Studio (e.g., C, C++, or Visual Basic). Groups were also informed that they could use any application desired for other parts of the task. Each user received $15 for participating in the study.

6.5.2 Procedure and Workspace

The experimenter met with recruited groups to go through an informed consent process and schedule a time for the programming task. The day before the scheduled session, the description of the task was e-mailed to the group. This allowed group members to begin thinking about possible solutions and how the task could be divided prior to the scheduled session, thereby maximizing the amount of time available for writing, testing, and integrating code.

The task was performed in our department's HCI lab. The workspace was configured such that users were sitting around a large conference table with HP tc4400 laptops as their personal devices and a large NEC 61" plasma display positioned nearby. The large display was driven by an independent PC. This configuration is representative of those commonly used in professional settings (see Figure 4.6).

The experimenter demonstrated the functionality of IMPROMPTU and provided time for the group to practice using it and ask any questions about the framework or task. Users reported they had read the task description prior to the study.

The group was given an hour to complete as much of the task as possible. Users then completed a questionnaire and participated in an open-ended discussion about IMPROMPTU. The session was videotaped, and the tapes were later analyzed to understand how the framework was utilized.

6.5.3 Results

We discuss results from our study, focusing on how groups utilized our framework, users' reactions to the features and functionality of the framework, and notable opportunities for improving the framework's usability and stability.

6.5.3.1 Use of the framework

Groups made extensive use of our framework throughout the task. For the hour that a group was engaged in the task, each group member had made available a total of 4 windows on average and typically had more than 1 window available at any given time. Also, each group member created a total of about 9 replications on average. Typically, replications were created in response to a new collaborative sub-task and were closed once the sub-task was complete. Each group member replicated at least one of their local application windows to the large display.

When asked about the framework's features and functionality, users provided many insightful comments about what aspects of the framework were useful and effective.

Users were very enthusiastic about the framework's support for different modes of collaboration. For example, one user stated that our framework allowed him to "work independently but easily and quickly be able to see [other] code that is being worked on and join in when needed." Another user stated, "It's very useful for big projects where each programmer likes to work on their own computer but at the same time they can give their code to their teammates and be able to [get] comments and help."

The ability to place task artifacts on the large display was also appreciated. For example, one user stated, "I found it extremely useful to be able to quickly glance up and see what my partner was doing and what files he was modifying". While another stated, "It's good for highlighting items and for clear communication. It provided a sense of security because I know exactly what they can see, there is no guess work".

Users also found value in being able to quickly replicate windows between personal devices. As one user stated, "it's definitely useful for looking at other's code – especially because I'm rusty with C++. I could easily see what [a team member] was doing without having to look over his

shoulder." Others commented that this type of replication would be especially useful when a large display was not available or when group members were working in separate offices.

Overall, the framework was primarily used in these contexts:

- *To request and receive assistance*. There were several instances during the task where a group member would request help or advice on their part of the code, e.g., to understand and correct a compiler error. In these cases, the framework was used to replicate the source code editor window between their personal devices, allowing the code and potential solutions to be more easily discussed. For example, it was common for each group member to interact with the editor window to reference specific lines of code, propose ideas, or capture agreed upon solutions.

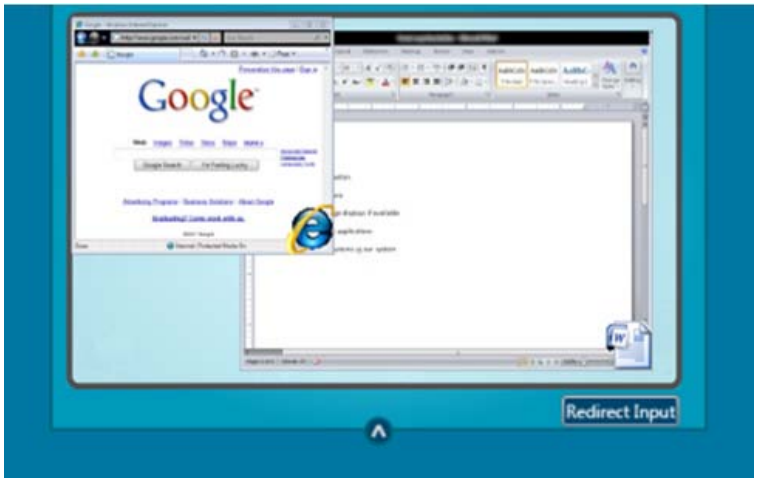
- *To create a repository of reference information for the task*. A group often found it necessary to research documentation, sample code, or existing code in order to provide necessary understanding for proceeding in the task. Since this reference information was generally beneficial to the entire group, our framework was utilized to place the relevant task artifacts on the large display. Also, multiple users contributed task artifacts to this repository.

- *To integrate individual contributions*. Once group members completed their individual tasks, they would attempt to integrate and test their pieces. In these cases, the group members would place their source code editors onto the large display and work together to perform the integration. For example, group members would advise each other on how to make appropriate calls into their respective parts of the code. In these cases, the ability for each group member to place their local application windows onto the large display was particularly beneficial.

Figure 6.7: The improved Collaborator Bar. Compared to the previous design, the expanded view automatically appears when the user mouses-over a user in the collaborator bar. To replace the summary view, the redesign adds a summary of the applications that has been made available by the user right below their image and name. This redesign removes an unnecessary step in the replication interaction and improves awareness.

- *To maintain awareness of group members' progress.* As group members worked on their individual assignments, it was often necessary to see how another group member was structuring her code. For example, one user wanted to ensure that he was using compatible data types while another wanted to verify cross-functionality between components. Our framework was utilized to provide this awareness in two ways. The first was for each user to place a replication of his or her code editor window onto the large display. This allowed the users to glance at the display to extract information they needed without interrupting group

(A)



(B)

Figure 6.8: The improved Shared Screen Dock. In both views (A&B), the user now is able to redirect local keyboard and mouse input to the shared screen. Returning input to the local device is achieved through a hot-key interaction.

members. A second method was for each user to *show* her code editor so that other group members could replicate it *as needed* to extract information.

Though not exhaustive, these examples demonstrate the groups wanted and were able to utilize the framework to quickly share task artifacts, transition between individual and (sub)group work, and maintain awareness. This was a very positive result given the relatively short duration of the task and that none of the users had previously used the framework or had prior knowledge of it.

Through building this initial understanding of use, we were also able to better understand the scope and content of the observational coding scheme that would be used in our

future filed study of IMPROMPTU. For example, we found physical movement and the different roles users placed on devices and people to be valuable measures in understanding how the MDE was used. Using these and other measures that we observed to be important, we developed a draft coding scheme. As we discuss in the next chapter, the scheme provided insight into how the use of IMPROMPTU impacted collaborative behaviors.

6.5.3.2 Opportunities for Improvement

Results from the study revealed several opportunities for improving the design of IMPROMPTU and similar systems. As a result, we have made the appropriate modifications to IMPROMPTU. These include:

- *Summary view in collaborator bar should be collapsed or removed*. Users overwhelming felt the summary view of the collaborator bar was not useful as users often bypassed the view to interact the fully expanded view. Further, users also felt the summary should be provided on the bar itself, without requiring a mouse-over action to get a summary of the applications a group member has made available. As shown in Figure 6.7, we modified the design of the interface to accommodate this design flaw.

- *Allow users to redirect local input to shared screens*. Although supported in previous interfaces discussed in Chapter 3, we did not initially include direct input capabilities with IMPROMPTU due to specific lessons learned in Chapter 4. From this user study, we found that allowing input redirection on the shared screen is a necessary feature. In our redesign, we added a redirect input button to both views of the shared screen dock (see Figure 6.8). This redesign provides input redirection to shared work surfaces (i.e. shared displays) but not personal work surfaces (i.e. personal devices).

- *Support cut-copy-paste operations involving replicated windows.* On several occasions, we observed users attempting to copy/paste information between

replicated windows or between a replicated and local window. Because the windows do not share a common buffer, the operation could not be performed, and was frustrating to users. To address this issue, the replications services was extended to also to maintain shared copy buffers (per user), and override the common key sequences to utilize this buffer rather than the local one when copy-cut-paste actions occurred within a replicated application.

In addition to these specific design updates, we also made numerous bug fixes to improve the stability of all three components of the framework. This usability study enabled the detection of these bugs, as most of the bugs occurred when the framework was being used in a complex configuration (e.g. significant number of replications across multiple devices). The study was thus instrumental in improving the design and robustness of the framework. However, the study did not investigate the use of IMPROMPTU in support of collaboration within an authentic, real world task domain. In the next chapter, we present a field study where we deployed IMPROMPTU within professional software development teams and studied how the framework impacted their collaborative practices over a period of several weeks.

Chapter 7

Evaluating the Effectiveness of IMPROMPTU to Support Co-located Software Development Teams

In the previous chapter we introduced our new interaction framework for MDEs called IMPROMPTU. The framework was designed to overcome many of the limitations of related frameworks, allowing users to more effectively leverage the benefits of an MDE to support for myriad collaborative activities. While a lab-based usability study revealed that users appreciated and used nearly all of the features and functionality of the framework, the nature of the study did not provide insights into how the framework impacted collaborative behavior. Understanding the implications of continuous immersion, as [9] state, would provide a much richer comprehension of the actual value and effectiveness of an MDE framework to support collaborative activities. Towards building this understanding, we performed one of the first field studies to investigate the use of MDEs within an authentic task domain.

In our field study, the framework was deployed for several weeks within teams who currently work in shared workspaces where personal devices and large displays were available. The teams used the framework to support their normal, work-related collaborative activities. The objective of the study was to learn what features of the framework were used and how frequently, how users' collaborative practices were impacted through the use of the framework, and users' reaction and feedback about using the framework for their collaborative activities. This evaluation methodology allowed us

to better understand the value our framework and produce lessons for improving the effectiveness of similar frameworks in the future.

As with our previous design and evaluation activities, we again chose to study software development teams for many of the same reasons discussed in Chapter 4. Additionally, because so many software teams currently work in shared workspaces, an additional advantage was that we did not have to force users to reconfigure and existing or relocate to a new workspace. This was significant, as it ensured minimal disruption to the groups' typical working practices and allowed for one less independent variable in our analysis.

7.1 Study Participants

We recruited development teams from Microsoft Corp., a U.S.-based software company. To recruit teams, we sent a study solicitation to a company mailing list. We followed up with teams that expressed interest by arranging short meetings to outline for them the goals of our study, discuss requirements for participation, and describe the data that would be collected. These meetings also allowed the researchers to establish an initial level of trust with the teams. We found this to be especially important to ensure that our presence observing their daily activities for several weeks would not be overly disruptive or uncomfortable.

Through this process we identified two teams that met our requirements and agreed to the conditions of the study:

Team Alpha is a group of 9 individuals (all male), including 3 developers, 3 testers, a technical writer, a program manager, and a software architect. The team works closely together to create sample applications and documentation that promote the use of Microsoft development technologies. When we observed them, they were actively working on a package to showcase Web 2.0 technologies.

All members of *Team Alpha* were physically co-located within the same workspace (see Figure 7.1). The space was also equipped with a large display to which all members had access. This configuration supported the team's heavy adoption of Agile practices [123]

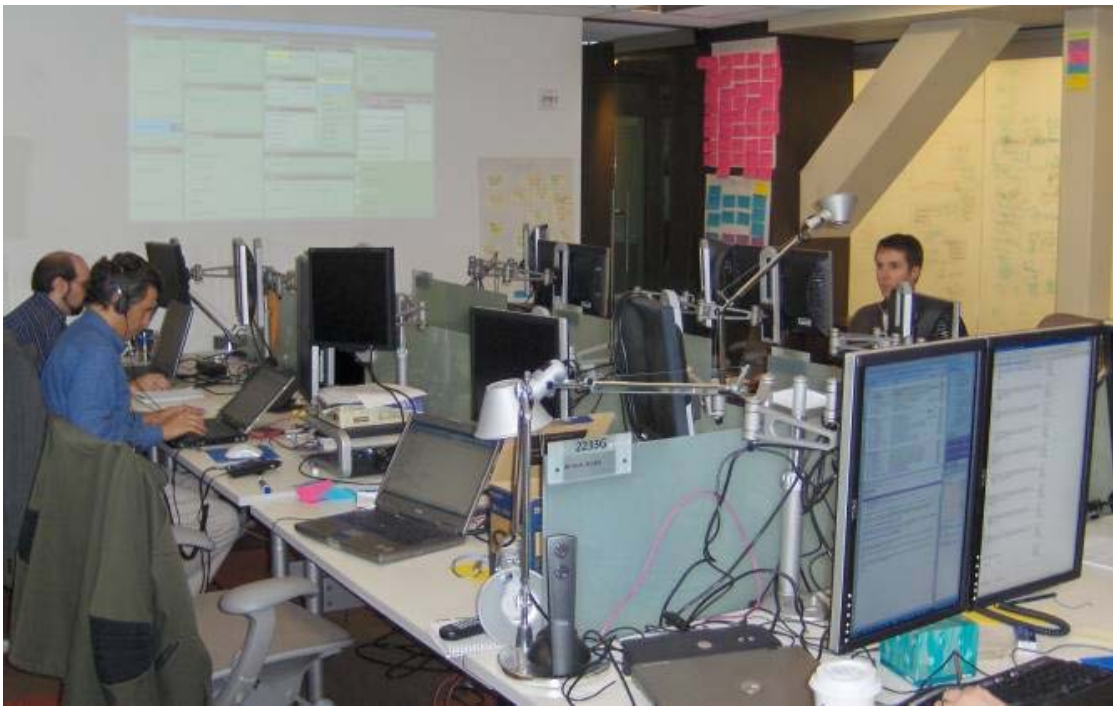


Figure 7.1: The physical space used by *Team Alpha.* Notice how each user has his or her own personal workspace and device. There is also a shared large display on the back wall of the space.

which promote activities such as paired programming, daily status meetings (a.k.a. "scrums"), and face-to-face communication.

Our second team, *Team Beta*, is a group of 6 individuals (2 female) including 3 developers, 2 testers and a program manager. *Team Beta* is a feature team that works on next generation software management tools. We observed this team in the late stage of their development process as they were performing final testing, bug fixes, and integration.

Unlike the other team, *Team Beta* had individual offices located within close proximity (just a few steps away). Team members worked in their own offices, but frequently traveled to each other's offices to collaborate, as well as to a group conference room that was equipped with a large display. Like *Team Alpha*, this team performed many tasks collaboratively, including editing, debugging, and review.

Each team member was provided with a software gratuity for participating in the study. Also, to provide incentive for filling out questionnaires (discussed in the next section), each submitted questionnaire served as an entry into a raffle for two Zune™ MP3 players, one for each team.

7.2 Procedure and Measures

The study was conducted over a three week period and used a split pre/post observation design. We observed the teams without IMPROMPTU for one week to gain a baseline of their current collaborative practices. The next two weeks were spent observing the teams using IMPROMPTU. Observations were split into alternating half-day sessions. That is, *Team Alpha* was observed in the morning and *Team Beta* in the afternoon. The next day, *Beta* was observed in the morning and *Alpha* in the afternoon, and so forth.

For each session, at least one observer was present in the workspace to code the teams' collaborative activities. For approximately half of the sessions, there was an additional independent observer who also coded the activities, which allowed for cross-validation. Two independent observers were used during the course of the study. These observers had over 20 years experience in behavioral data collection, were *not* researchers working on this project, and were not directly affiliated with the University of Illinois at Urbana-Champaign or Microsoft Research.

7.2.1 Observation Data

Many coding schemes exist for the capture and categorization of user behavior, and our investigation into creating a scheme led us to consider four widely used schemes. Interaction Process Analysis (IPA) [14] enables experimenters to capture and categorize both task-oriented and social-emotional leadership roles within a group. Gutwin et al.'s mechanics of collaboration [58] codes the basic mechanisms of teamwork. Olson et al.'s work on analyzing collaboration [99] and d'Astous work on collaborative programming [38] both provide a rich set of categories within which to classify activities that take place during collaborations within software teams.

However, these existing schemes did not cover the complete set of activities that we found to be essential to our study and that would be effective for our workspace and group configurations. Further, many schemes lacked classifications that could be easily quantified and compared. As a result, we developed our own new coding scheme that was a synthesis of the most relevant parts of existing schemes and our basic requirements. Our scheme consisted of 6 primary categories and is detailed in Table 7.1.

Instances of collaborative engagements that could be externally observed were coded. An instance would be signaled by initiation of physical movement to each other's work areas and/or verbal communication. Instances were considered complete when the verbal communication ended and/or users returned to their own work areas. To further limit the complexity of the coding scheme and the required analysis, we only coded activities that were "work task related." That is, only activities that were directly related to a work item. Non-work related social collaborations (e.g. catching up on weekend personal activities) vary widely based on personality, relationships, and environmental factors, and were excluded from collection. Because the focus of the study was to understand the impact of IMPROMPTU on the collaborative practices as they directly relate to group-based software development activities, we felt excluding non-work related activities would allow us to best satisfy our study goals. Further, to gain the trust and access to the teams studied, we specifically agreed not to collect non-work related collaborations.

A brief example will illustrate how the scheme was applied. Suppose one user physically moves to another's device and they work together to solve an error in the code. Along with modifying the code, they consult discussion forums and API documents to explore solutions. This instance would be coded as Evaluation and Explore Alternative Solutions under Activity; Edit and Debug/Test under Domain; Move to Personal Device under Physical Movement; and Single Personal Device under Use of Devices.

We created a semi-automated Excel spreadsheet for collecting observation data. The spreadsheet (shown in Figure 7.2) allowed the coders to quickly annotate a collaborative engagement. The spreadsheet also automatically set the start and end times of the collaborative engagements. Synchronized system clocks across the independent coders and the instrumented tools used to collect framework usage data allowed us to easily line-up and synchronize the data for later analysis.

Category & Classification		Classification Definition
Activity	Cognitive Synchronization	Establish a common representation of a given subject.
	Evaluation	Judge the value or give opinion on a particular subject.
	Explore Alternative Solutions	Proposal of new solution to the topic under study.
	Conflict Resolution	Discussion of conflict and potential resolutions about a given subject.
	Management	Planning of future or ongoing work sessions.
Domain	Edit	Generating new code or modifying existing code.
	Debug/Test	Stepping through code or testing features of the software. Small modifications may be made to further uncover software defects.
	Review	Reading and discussing the code. Often occurs when trying to understand never before seen code or preparing to commit code into the repository.
	Reference	Consultation of auxiliary information such as API documentation (e.g. MSDN website or reference book), sample code, or personal notes.
	Design	Brainstorming/planning new features. Often occurs before the code editing. Could also include creating or updating design documentation.
Physical Movement	No Movement	Collaborators do not physically reconfigure.
	Move to Personal Device	Collaborators relocate to work around a single personal device.
	Move to Large Display	Collaborators relocate to work around a single large shared display.
	Other	Collaborators relocate to configuration other than those above (specific configuration should be noted).
Use of Devices	Single Personal Device	Collaborators perform their activity using only a single personal device.
	Multiple Personal Devices	Collaborators perform their activity using multiple personal devices.
	Personal Device(s) and Large Display	Collaborators perform their activity using both personal devices and a shared large display.
	Large Display Only	Collaborators perform their activity using only a single shared large display.
Time	Length of collaborative engagement	The beginning of a collaborative engagement was denoted by two or more individuals initiating a verbal conversation that is related to work topics and/or physically relocating within the environment. The end of an engagement was denoted when the verbal conversation had ceased and/or individuals returned to their personal work areas.
Size	Number of individuals in collaboration	The total number of people that participated in the collaborative engagement.

Table 7.1: Category and classification definitions used for coding observed instances of collaborative engagements.

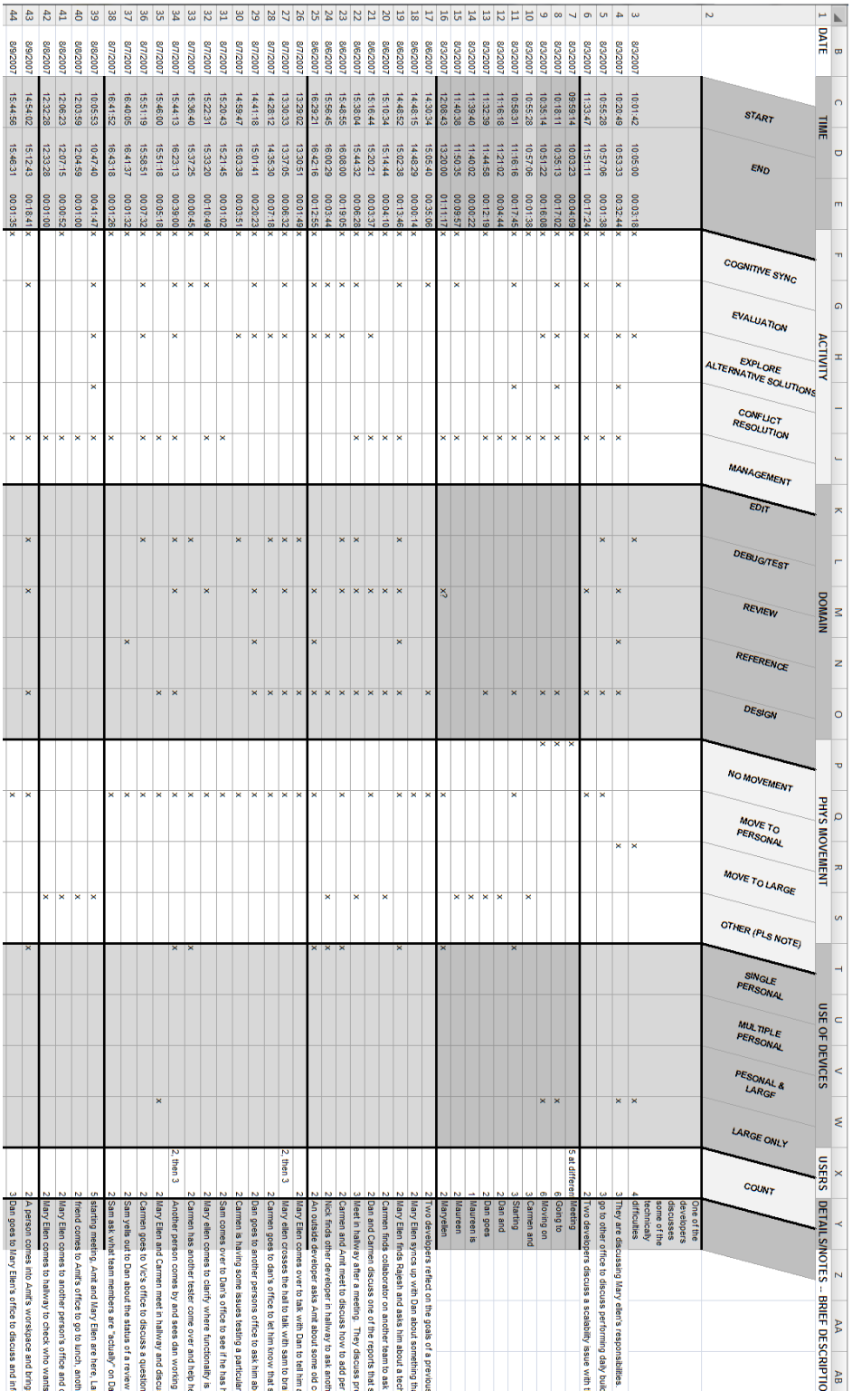


Figure 7.2: A screen shot of the semi-automated Excel worksheet used by the coders to collect observation data during the field study.

To validate the coding scheme, we performed a two hour pilot observation session with the independent observers. Following the observations, the researchers and observers reviewed the data collected, discussed and resolved coding differences, and made small modifications to the coding definitions to help reduce future coding differences. For the actual study, inter-coder reliability was measured by sampling 10% of the coded data and computing Cohen's κ [35], a common measure of coder reliability. All but two categories (Review and Edit under Domain), had Cohen's κ≥ 0.76, which indicates a good degree of agreement (see [11] for details on interpreting κ values). The remaining categories had lower Cohen's κ (0.25 & 0.46, respectively), but this was due in part to having unbalanced categorization frequencies. A further test of raw agreement showed that these categories had > 80% consistency, and we believed this represented sufficient agreement.

7.2.2 Instrumented Data

We instrumented our framework to log usage data. This was done to understand which features were used and how often. This data included which applications had been shared or shown and for how long, which applications had been replicated, to which device they were replicated and for how long, how often input redirection was used, and how often users interacted with the Collaborator Bar.

7.2.3 User Feedback

Finally, we collected user feedback in two forms. First, we asked users to complete a questionnaire that probed their use of the framework for a recent, meaningful collaborative activity. Questions included explaining the motivation for the activity, who was involved and their role, which aspects of the framework were used, and how it affected the overall activity. Each team member was asked to complete the questionnaire every other day using an online form.

In addition to the questionnaires, we performed a 30 minute group interview with each team at the end of the study. We asked the teams about their overall experiences using the framework, what they felt were its strengths/weaknesses, and for recommendations on how it could be improved.

7.3 Results

In this section, we describe how the teams utilized our framework, provide results from the instrumentation data, and discuss impacts on existing collaborative practices.

7.3.1 Use of IMPROMPTU

During the study, our framework was leveraged to perform myriad collaborative tasks. These tasks included providing opportunistic assistance in solving complex compiler errors, working closely together to integrate different users' code into a single solution, reviewing API documentation, and brainstorming designs for new features. In support of these tasks, users leveraged our framework to show or share a wide variety of applications, including source code and document editors, communication tools, and Web browsers. The applications were replicated between personal devices as well as placed on the large displays. A few detailed examples will help exemplify the use of the framework. In these examples we use pseudonyms to protect the privacy of our participants.

In one instance, we observed John, a developer from *Team Alpha*, encounter a compiler error while working on his code. After spending a few minutes attempting to solve the error, he requested assistance from Stan, another developer on his team. John used the Collaboration Control to set his source code editor to *share* and Stan replicated the window by dragging its thumbnail from John's area within the Collaborator Bar. Stan interacted with John's code editor to review and understand the problem, used the telepointer to highlight segments of possible problem code for John, and offered verbal suggestions on how to proceed.

In another instance, the manager on *Team Beta*, Felix, needed to send a status report to his boss. Because he was unsure on technical details, he asked Susan, a developer on the team, to assist. Felix shared his document window, and Susan replicated it on her device. A verbal communication channel between the two was established via the office phone. Felix and Susan were able to jointly compose the report, each providing content that they knew most about.

In a similarly structured task, Steven and Greg, two developers on *Team Beta*, used the framework to review and integrate Steven's code into the source repository. Greg replicated a code editor and a separate integration tool that Steven had *shared*. Greg spent a few minutes reviewing Steven's code, while Steven multitasked between answering an occasional question from Greg and his email. When Greg finished reviewing a section of code, the two rejoined working in the code editor, made modifications to the code, and then moved on to the next section.

Aside from peer-to-peer tasks, the teams also engaged in group-wide tasks. Members of *Team Alpha*, for example, conducted a team-wide design session. In the session, the lead developer replicated an architectural diagram from his personal device onto the group's large display. Using the diagram as a shared visual reference, the team stood near the large display to discuss the content of the diagram and its implications on their immediate and future work.

In another team-wide collaboration that was focused on feature planning, a developer replicated a note taking application onto the large display and the team took turns adding content to the list. Team members added content by interacting with replications of the note taking application that they had also created on their personal devices.

Finally, one of the more inventive uses of the framework that we observed was to place information on the large display to *passively* attract the attention of team members. For example, in *Team Alpha*, a developer placed a trade news article discussing a competing product on the display to entice discussion and comments from team members.

Though not exhaustive, these examples demonstrate that teams did indeed utilize the framework to perform many different types of collaborative tasks. Users were able to quickly and easily make task-related artifacts available to the group, without disrupting the natural flow or pace of the collaboration. For example, users could, in most cases, quickly transition from individual work to joint interaction with an application without having to physically move or reconfigure devices. For example, as one user commented,

he appreciated "the ability to have multiple team members actively manipulate the same application."

Users also appreciated the ability to share windows while maintaining the ability to multitask with other applications. As one user wrote, "I like that I can be actually working on another team member's machine (reviewing code, editing a PowerPoint, editing a word doc) in a hidden window (off to the side) while not blocking them from doing other work."

Users could also quickly make information available for group review and comparison through the ability to place applications from multiple devices onto large displays. One user said the most useful piece of functionality was "the ability to easily display items [from] the desktop to [the large display] where they could be viewed by others." Another said it was useful because "you could see that someone was collaborating, and might choose to jump in."

7.3.2 Instrumented Measurements

We collected over 1035 hours of system use across the 15 users. The amount of use by each user was highly varied; the overall mean length was just under 74 hours (SD=64h04m). The distribution, though, was bimodal; a subgroup of 7 users used the framework for a mean of 16h21m (SD=3h46m) during the course of the study, while the remaining 8 had a mean of 131h38m (SD=17h24m). These results exemplify two types of use of the framework. The less frequent users would tend to only launch the framework when it was needed (e.g., to initiate or support a replication) while others would launch and leave it running throughout the entire workday (e.g., to make applications available to facilitate awareness for team members).

96 applications were made available to the group in total; 74 were *shared* and 22 were *shown*. On a per user level, mean *shares* was 6.73 (SD=10.1) and mean *shows* was 2.0 (SD=2.1) for the period the framework was deployed. This result illustrates that users typically only made applications available to the group when there was an immediate need. It also indicates that the owner of an application deliberately considered the control other members should have over it.

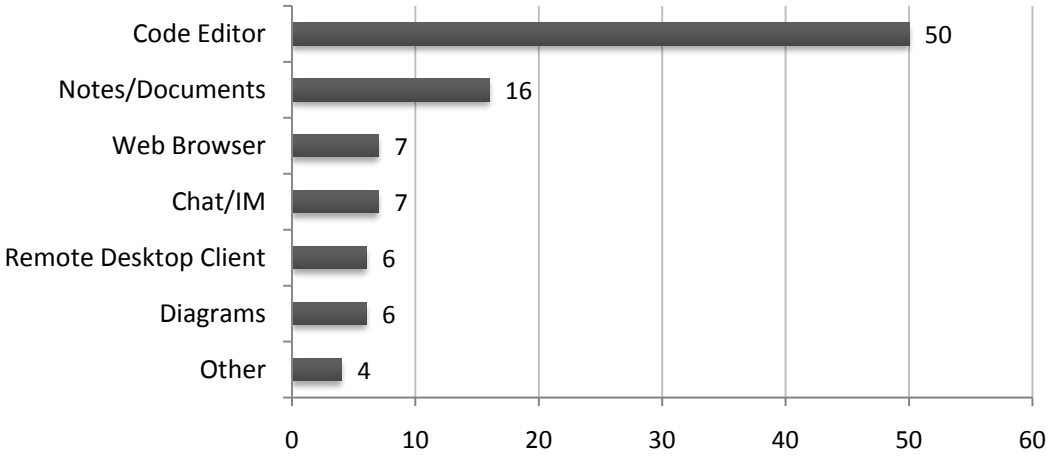


Figure 7.3: A breakdown of the type of applications that users made available to their group using the framework.

Figure 7.3 shows the distribution of application types made available. Given the task domain, it was expected that the majority of applications made available would be source code related. But, many other applications were also made available including instant messaging (IM) applications, diagramming tools, and Web browsers. This highlights the need for MDE frameworks to support myriad applications.

Of the 96 (~80%) applications that were made available, 77 of them were replicated. The mean time that an application was replicated was 13m38s (SD=23m33s). 19 (~25%) of the replications occurred in collaborations in which more than one application was replicated at the same time. This functionality was leveraged, for example, to compare information that was distributed across several devices.

At the device level, the median number of applications that were replicated *from* and *to* each device was 6.00 (SD=4.0) and 2.50 (SD=9.7), respectively. The imbalance is due to the heavy use of the large displays (35/77, or 45.5% of the replications). This shows that both personal-to-personal and personal-to-large display replications were utilized.

There were 898 Collaborator Bar interactions, a per user mean of 68.9 (SD=41.9). This shows users were leveraging this component for establishing replications as well as for acquiring awareness of others' current tasks.

Finally, the users rarely used the input redirection support (only 3 instances logged). This result indicates that users in our study strongly preferred to perform input actions via replications on their local device, rather than on the shared display. This result could partially be influenced by the work styles and task domain studied. Nonetheless, if future studies confirm similar low use and utility, it would be strong evidence to push this functionality into the background (requiring a heavier interaction to initiate and freeing up resources in the interface that could be assigned to more frequently used functionality).

7.3.3 Impact on Existing Collaborative Practices

We collected 125 hours of coded observational data over the three week period of the study. 50 hours of observation were conducted prior to the introduction of our framework, and 75 hours were conducted with our framework. The coded data is summarized in Figure 7.4.

For Physical Movement, a change was found in the number of movements to a single personal device in *Team Beta*. Movements dropped from 66% to 40%. No differences were found for *Team Alpha*. Analysis did not reveal significant differences in the other categories.

Despite not having detected changes in most of the categories, we do not believe this is a negative result for two reasons. First, the framework was specifically designed to support existing collaborative practices, not to necessarily change them. Second, the use of the framework was studied within teams who had well-established working relationships (e.g. the teams had been working together for several years). The fact that the teams used the framework in support of many tasks without a detectable change in observed behavior suggests that the use of the framework was integrated into their existing collaborative practices. Indeed, as we found in the interviews, users stated that a central benefit of the

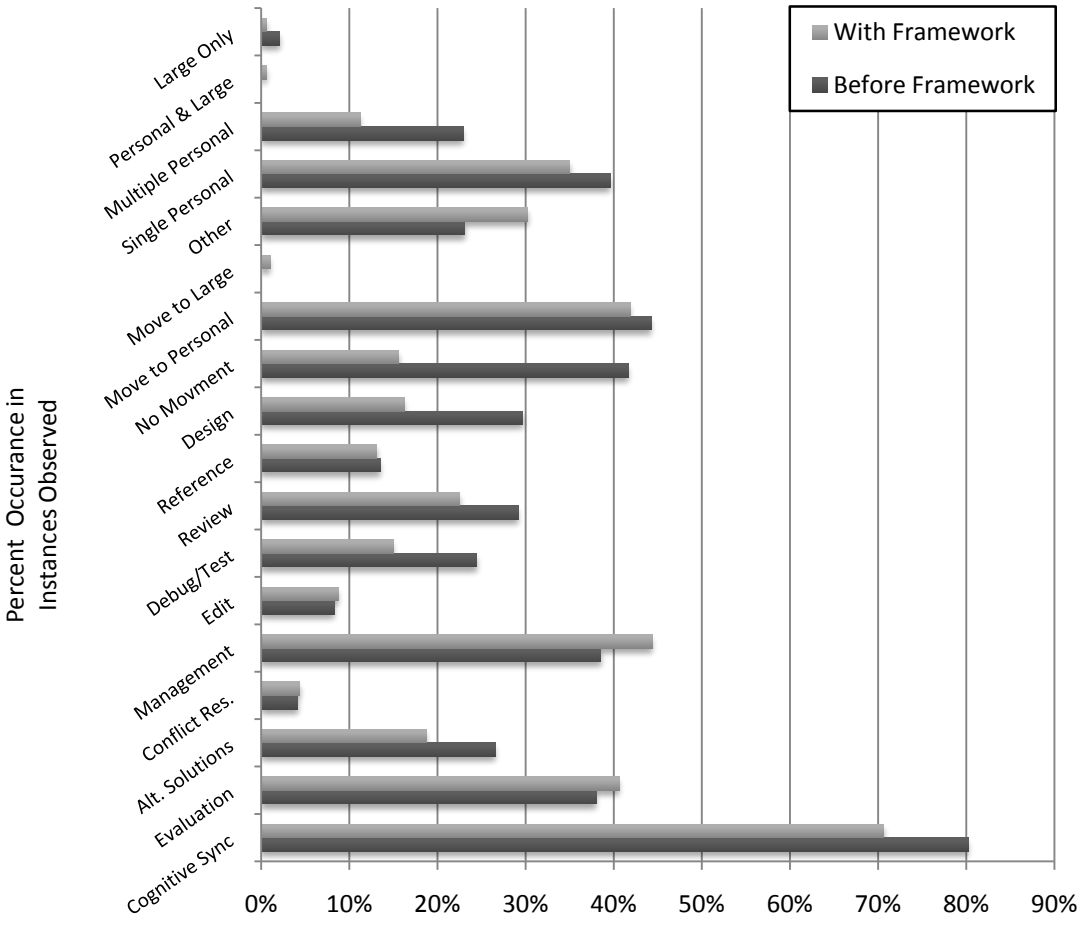


Figure 7.4: This chart summarizes the observation data gathered before and after the framework was deployed. Results are reported as the percentage of total collaborative instances observed per condition (194 before/160 with framework).

framework was that it allowed new opportunities for collaboration within their existing group practices.

This result does not necessarily mean that changes in group behavior would not occur. Rather, it suggests that such changes, if they exist, would occur gradually over a longer duration of time. We plan to further investigate this issue in future studies.

7.4 Discussion

Overall, our study showed that groups did find value in using our framework (in an MDE setting) to support software development. This is evidenced by the wide range of tasks performed and applications shared, and by many users stating during the group interview that they would want to continue using the framework. Several users also stated that our framework allowed the large displays to be more tightly integrated into their work practices, as their use could now be shared without configuration overhead.

Teams utilized almost all of the features of our framework, but its usage frequency was relatively modest overall. We attribute this usage behavior mostly to the task domain, as users were often focused on their own individual efforts. The value of the framework was thus derived not from its frequency of use, but in its utility in supporting specific instances of collaborative engagements. Users expressed, however, that the framework may have been utilized more frequently in situations where collective understanding of the technical content is low, e.g., during project planning or initial coding, resulting in more information being shared.

The use of our framework was studied in one complex task domain, software development. This decision was made because it would allow us to study the actual benefits and use of this type of framework in an authentic setting, filling a large gap in the current literature on MDEs. Groups in other domains that share similar work processes, e.g., to serendipitously share task information, transition between individual and joint work, and collectively manipulate content on large displays, may benefit from and/or use our framework in similar ways. Collaborative design and creative writing offer potential domains in which to further study the use of our (or a similar) framework.

Though studied in one domain, results from our study did reveal opportunities for improving the design of MDE frameworks in general. One improvement would be to enable the owner of a window to know the status of the associated replications on other users' machines. For example, users wanted to know when a team member was actively working within a replicated window or if it had been moved out of focus or minimized,

as this would allow them to better time appropriate transitions to (sub)group work. One possible solution would be to provide visual cues of replication status within the title bar or other decoration of the source window on the owner's device.

A second improvement would be to merge the framework's Collaborator Bar with the contact lists already available in existing communication tools such as e-mail and chat. This would reduce the need for maintaining separate collaborator lists and free up valuable screen real estate. It could also enable cross-utilization of collaboration tools. For example, an ongoing group chat could easily expand, if desired, to a collaboration involving several application windows shared among the users' displays.

We also found that the framework should enable users to manage sharing options from any device within the MDE. For example, once engaged in a task at another user's work area or large display, a user may want to access applications running on her personal device. The current design of our framework requires the user to move back to her device to set the appropriate permissions. A possible solution would be to allow the user to enter a password into her representation on the Collaborator Bar to set all running applications on her local device to be shared (or a subset based on defined rules), allowing the desired applications to be available

Chapter 8

Discussion and Future Work

In this chapter we discuss some of the issues related to the design and evaluation of our framework within the context of the broader HCI and CSCW research domains. We also discuss future directions with this work.

8.1 Observed Changes in Collaborative Engagements

Through the design, implementation, and evaluation of IMPROMPTU, we gained many important insights towards understanding how MDEs impact collaborative work. Analysis of the observational data showed that utilization of the framework did not significantly shift users' collaborative practices. While there are many possible reasons for no change, we attribute the result to indicate that the framework enabled the group to effectively work together using their existing practices; i.e., it did not force them to work in ways that were unfamiliar and/or distracting from accomplishing the collaborative activity. Feedback from the study participants reinforced this as a positive outcome. Many participants indicated that their current customs and practices were fine tuned over hundreds of hours working together and that they would likely not use any tools or systems that required the group to immediately change the ways in which they work.

This result raises the broader question about whether or not a groupware system needs to invoke a change in how users collaborate to be considered effective. It is certainly true when users' current collaborative practices are flawed and the specific purpose of the groupware is to improve such practices. However, with groupware like IMPROMPTU,

where the goal is to improve the *existing* collaborative practices, immediate and significant change in collaborative practice is not likely a positive outcome. Grudin argues that a significant challenge in designing this latter category of groupware is to create solutions that respect and uphold group and social dynamics, allowing users an "unobtrusive accessibility" to using the features of the software to perform collaborative activities [55]. The effectiveness of such groupware should thus be measured on its ability to be adopted by users and integrated *into* existing practices. The results of our study support such a result with IMPROMPTU.

Results from the study also showed that even modest use of the framework had significant value to its users. For example, even though users in our study only averaged about two replications per day, those replications were used to significantly improve how the collaborative task was performed. Such tasks for the domain studied included assisting in application debugging, reviewing documentation, planning future application features, and more.

Past studies of MDEs were fairly limited to users performing short, simple tasks (e.g. a sequence of application replications) and, as a result, focused on associating effectiveness to the quantitative use of the MDEs' features. While this approach may be sufficient for lab studies, the qualitative use of the MDE often outweighs quantitative use when trying to understand impact on *authentic* use [9, 55]. Our study was one of the first to provide insights about the value of MDEs that were derived not only from frequency of use, but also from analyzing the quality of user interactions.

8.2 Value of Principal Features of IMPROMPTU

Based on the results of our study, the ability of the MDE to effectively support users' collaborative activities were due in large part to new functional and usability innovations provided by IMPROMPTU.

8.2.1 Independent Shared Input Processing

A significant technical advantage of IMPROMPTU is its ability to route remote input to shared applications without overriding the local input actions on the host device. Unlike existing systems and tools that require input focus to be maintained in the shared application, IMPROMPTU enables users to freely switch between shared and non-shared applications. This affordance was leveraged by users in our field study to support more effective multi-tasking when working on collaborative tasks. As an example, we often observed users leveraging this affordance to perform group debugging sessions. In these activities, one user (the current person that had the code checked out of the source repository) would share his code editors with peers. The developers would then take turns interacting with the shared code editor to explore solutions. While some edited code, other developers would create private instances of web browsers and project management tools on their local devices to research other potential solutions independently, including the user that is the owner of the shared code editor. This example illustrates IMPROMPTU's ability to enable users to seamlessly use, and transition between, personal and shared applications. Such an affordance is not provided by existing screen and application sharing tools.

8.2.2 People-Centric User Interface

A key innovation of IMPROMPTU's interface is its people-centric interface that enables quick identification of participating collaborators and the applications which they have shared with the group. With the interface users can also quickly, and with relatively low interaction overhead, instantiate replications of shared applications. In our field study, these design elements were shown to have high value to users as they facilitated users' ability to easily engage in ad-hoc, serendipitous collaborations – situations where introducing a shared piece of information at the right time greatly aided the collaborative task.

In our study, we observed many instances where the value of quick access to users and applications was evident. For example, users found IMPROMPTU facilitated their ability to provide quick coding assistance on bugs, compiler errors, proper API syntax, etc. In

these cases, users would often keep their code editors shared so that, when help was solicited, their peers could quickly establish a local replication of the editor. As many participants in the study commented, this enabled the assisting user to see and understand the specific context related to the issue which enabled them to provide better informed recommendations.

8.2.3 Application-Level Multi-Source Sharing

Another key advantage of IMPROMPTU is its ability to allow users to replicate applications from multiple sources *at the same time*. Participants in our study found this especially useful when leveraging the large display as a workspace for comparing and combining information from multiple users. For example, in their daily planning meetings, team members from one of the groups studied would place information from several independent devices on the shared display at the same time to aid discussion. These items ranged from management tools summarizing open bugs to even quick design sketches. While many existing solutions support multi-destination sharing (i.e. one-to-many), IMPROMPTU is the first sharing framework to support both multi-destination and multi-source (i.e. many-to-one). This distinction further illustrates the versatility of the framework.

8.2.4 Negotiated Sharing Model

The push-then-pull negotiated model for sharing applications was also shown to be of high value to users. Specifically, post-evaluation discussions revealed that users appreciated the fact that they could make an application available to the team without forcing the application to immediately appear on other users' devices. This was useful, according to users, because it respected the natural flow of engaging in a collaborative task. As one example that we observed, to gain feedback on a proposed project timeline, a lead developer made the application available to the rest of the team, and sent an IM announcement to the rest of the team requesting that they review the timeline and make comments or adjustments before they started their planning meeting later that afternoon. This allowed the members of the team to view and interact with the shared application when it was convenient for them. In contrast, existing solutions provided little control for

specifying if and where shared applications appear on replication consumer's local device. By enabling more control for the replication consumer, IMPROMPTU upholds users' ability to maintain control over their private, individual workspaces.

8.2.5 Beyond Simple Screen Sharing

The overall value of IMPROMPTU lies in its ability to support important principles of effective groupware design. For example, the framework's independent input processing is crucial to supporting multiple, simultaneously occurring tasks and fluid transitions when switching tasks (see Chapter 2, Section 3 for further details). Existing tools do not support this capability, and as such, fail to realize this principle.

Additionally, IMPROMPTU was adopted as the preferred sharing tool, even when commercial tools (e.g. Microsoft LiveMeeting) were installed and available to the team. Participants in the study indicated their preference for IMPROMPTU because of its ability to support myriad modalities of collaboration, another core groupware principle (see Chapter 2, Section 3). Users were supported working tightly-coupled (e.g. working jointly to edit source code), loosely-coupled (e.g. share documents for review while working on independent tasks), or somewhere in between (e.g. work collectively to solve a bug in code while research solutions privately).

As we discuss above, satisfying these important principles proved to be critical in supporting the collaborative activities of our study participants. As a result, it is unlikely that performing a similar study using only a basic screen sharing tool would have produced the same results. In future longitudinal studies, we intend to investigate and show the differences between IMPROMPTU and other, more simplistic tools.

8.3 Replication Model

A central design decision in developing our interaction framework was to use a replication model. As discussed in Chapter 5, the purported benefit of this model is that it would allow any off-the-shelf application to be shared across devices. This gives the ability for users to leverage single user applications that they are already familiar within collaborative activities. Indeed, results from our field study (Chapter 6) showed that this

design decision was justified, as users chose to leverage this benefit to share a wide variety of applications; ranging from code editors, Web browsers, to document editors. Another benefit is that the model enables the interaction context of applications to be maintained when replicated (e.g., keeping breakpoints and stack data in a debug tool). This proved to be valuable, as it minimized the overhead when transitioning from individual to group work and vice versa.

There are, however, several important limitations to the replication model. A main criticism of this model is that users cannot independently manipulate the view of a shared application. For example, the replication model does not allow two users to collaboratively edit a word processing document while working in separate sections (e.g. one person edits the introduction and the other the bibliography). In some cases, users not only share the same view but also the same modality. For example, if one user sets the application to a specific modality (e.g. caps lock or overwrite) all other users must work in that modality as well. This high degree of coupling can potentially limit users' collaborative effectiveness.

Alternative application sharing approaches, such as the approach the creators of CoWord [151] followed, allow for applications to be shared while preserving the ability for users to maintain independent views. In these solutions, specialized applications or frameworks are needed to centrally coordinate and synchronize independent applications on each device. While theoretically possible, there is the significant cost of developing these specialize applications and frameworks for *each* application that is used in support of collaborative activities.

From our field study, we did not find any evidence that lack of independent views was a major deficiency of our framework. This result is likely due to that fact that the framework was used mostly to support serendipitous, short-lived collaborative engagements. In these cases, it was often useful and desirable for users to maintain the *same* view. Also, we found that when independent views were desired it was often because the users had transitioned back to performing individual tasks. In these cases,

users would just default back to using their existing set of single user applications and tools.

User satisfaction with the replication model could certainly be influenced by their working practices and the domain in which they work. Future studies that investigate the use of MDE frameworks in other task domains should certainly examine the sufficiency of the replication model.

8.4 Generalizability

A key factor in choosing the software development domain was its representativeness of other collaborative domains. We examined the work practices of software development teams and compared their practices to standard classifications of collaborative practices (e.g. [14, 38, 58, 99]). Software developers engage in a wide range of practices that included, but are not limited to: building common ground, reflection, reinterpretation, task management, and conflict resolution. Based this analysis, our framework would likely be useful for other task domains that share similar classifications. Such domains might include collaborative writing, group planning, creative design, and more.

Conducting our field study of IMPROMPTU allowed us to further understand its generalizability. We found that a key strength of the framework's design and functionality is its ability to enable users to quickly bring in digital content to a collaborative task (e.g. quickly share a source code editor, document, or website). This was especially useful for software developers as they work with a large volume and variety of digital information artifacts.

However, we also recognized that this is not always the case for all users and task domains. For example, users working on a collaborative writing task would rarely need to share anything more than a single shared document. Also, some task domains may require specialized tools to facilitate a specific collaborative process. As an example, industrial designers performing a brainstorm session would likely need tools that better support idea classification and organization.

To support the collaboration needs in other task domains it is likely that modifications would need to be made to the design of IMPROMPTU. For example, to support design sessions, the collaborator bar could be changed to organize the workspace by idea categorization rather than by people. Similarly, specialized collaborative applications may be needed to work alongside IMPROMPTU. For example, tools like TEAM STORM [61] that are designed specifically to support the creative design process.

Even with these modifications and additions, many of the design lessons derived from creating IMPROMPTU (and this dissertation more generally) would still have applicability. For example, MDE solutions for *any* task domain still need effective mechanisms for controlling access to shared artifacts, maintaining awareness of artifacts and collaborators, and interacting with shared displays. These are principles that underlie almost all forms of face-to-face collaboration.

8.5 Large Scale Deployment

Large scale deployment of IMPROMPTU could allow a large number of end users to realize the benefits of MDEs for everyday collaborative activities. However, a large scale deployment of the framework would require several logistical, technical, and legal obstacles to be successfully overcome. One important logistical obstacle is providing coordination servers that are publicly accessible. This is a non-trivial problem, as it would likely require several load balanced servers to be configured and maintained on an ongoing basis. Thus, it raises questions about where these servers are located, who controls and maintains them, and how their owners recoup the cost of operating them. One potential solution is for an independent entity to provide access to community servers while charging a fee or tax to establish collaboration session or for creating replications. Supporting IMPROMPTU within a single organization is less difficult, but still requires a responsible party to setup and service the organization's coordination server.

Another concern with large scale deployment is the remaining technical challenge of IMPROMPTU not supporting *all* off-the-shelf applications. The way the framework is

implemented enables replication of applications that are built on the standard windowing APIs provided by the Microsoft Windows SDK. Most, but not all Windows applications are built using this toolkit. For example, the popular Firefox web browser does not respond to IMPROMPTU's remote input framework. This is because Firefox is built using a non-standard windowing API that does not place input on Window's Message Bus, preventing the framework from overloading input commands to the application. It would be ideal for an alternative input solution to be developed for these non-standard applications for which IMPROMPTU can seamlessly switch to when necessary. In this way, the framework would be more accommodating to these non-conforming applications.

Finally, a relatively major obstacle is the implications application sharing has on the licensing and terms of use (ToU) agreements of commercial software. Most licenses and ToU only allow the application to be installed and used on one computer per license. Thus, when a commercial application is replicated using IMPROMPTU, it is likely violating the application's conditions of use. Overcoming this problem is largely legal and economic – software vendors will need to update their licenses and ToU to specify replication privileges. These changes could range from users being required to purchase additional licenses to a set number of replications being built into the terms of use. It would then be ideal if the applications could communicate with the framework what the license and ToU allow.

8.6 Support for Distributed Collaboration

While IMPROMPTU was designed specifically to support co-located collaboration (i.e. same time, same place), many of its features and benefits could potentially be applied to supporting synchronous distributed collaboration (i.e. same time, different places). However, there are several limitations that would first need to be addressed before IMPROMPTU could be used in this capacity. While a rigorous user-centered design process is likely necessary to fully understand these limitations, our own experiences designing and evaluating IMPROMPTU has highlighted several necessary improvements and additions. These include:

- *Verbal Communication Channel*. The design ofIMPROMPTU assumes users have a verbal communication channel between collaborators. As we and others have found, this channel is imperative to supporting effective collaboration. A challenge in supporting effective distributed collaboration is being able to recreate this affordance. This could be provided through telephone conference integration or more advanced voice over internet protocol (VoIP) tools built into the framework. Use of instant messaging (IM) channels could also be used.

  The inclusion of a voice or IM channel into the framework must be carefully designed so that it preserves the framework's low interaction overhead. The establishment and maintenance of these communications channels must not distract from the collaborative task.

- *Improved Awareness Mechanisms*. A significant design challenge with all distributed groupware is providing appropriate and adequate awareness. Distributed use of IMPROMPTU raises three main awareness needs: awareness of collaborators and their availability, awareness of shared artifacts, and awareness of ongoing collaborations or sub-collaborations.

  Many existing solutions proposed by the community could be easily included into the design of the framework to provide improved awareness. For example, the collaborator bar in IMPROMPTU could provide visualizations of availability that are similar to those provided in OpenMessenger [26]. If the distributed collaboration is between multiple MDEs, then the large displays could be used to show visualizations of workspace activity. For example, using a large display to show FASTDash [25] would provide a persistent visualization of shared artifact state. Another display could show visual summaries, such as a collage of the people and applications being shared, to promote awareness of ongoing collaborations.

- *Improved Network Latency Tolerance*. The current implementation of the framework assumes users are co-located and connecting over a local area network. Because the network latency is small in these environments,

IMPROMPTU does not make special accommodations. However, latency becomes larger and more variable across wide area networks like the Internet. With IMPROMPTU, large network delays could result in remote users experiencing long delays when interacting with replicated applications. This delay could introduce confusion and frustration, especially when two or more users desire to work simultaneously within a shared application.

To overcome this limitation, the framework could exploit better compression techniques for sending screen and interaction data. For example, the streaming module could send only the pixels that have changed since the last screen update. Additionally, we could employ existing groupware-optimized compression techniques like GMC [56] to improve performance. Also, IMPROMPTU's user interface could provide feedback to the user to alert them of the delay and help mitigate its impact on the collaborative process. For example, a replicated application window could fade to grey when delays are detected. Others participating in the replication could also be notified that some users are experiencing delays. Such feedback could help groups stay synchronized.

8.7 Future Work

There are several future directions of research related to this dissertation that we are interested in exploring. We briefly summarize our immediate future work in this section.

8.7.1 Longitudinal Field Study

The field study performed in this dissertation provided many insights into the use and utility of MDEs to support authentic collaborative tasks and was an important first step in moving the field of MDE research out of the laboratory and into the real world. In fact, the foundational understanding of authentic MDE use provided by the study generates new questions about their use. Questions concerning the impact of using the framework and an MDE over a long period of time are now of particular interest. For instance, it would be interesting to understand if various features of the framework are used less or more often after continued use, and if these changes impact how users perform tasks.

A multi-month longitudinal study could provide these and other interesting insights. In addition, a longitudinal study would provide even stronger evidence and understanding on the validity of our design decisions. There are of course several logistical obstacles to accomplishing such a study; e.g. finding groups (and companies) that are willing to commit to a long term study. More important though are the non-trivial issues related to the experimental methodology. For instance, determining what behaviors should be studied, what measures should be taken, and determining the length of the study. In the future, we intend to explore these challenges in a longitudinal study of IMPROMPTU.

8.7.2 Investigating Use in Other Task Domains

We are also interested in further investigating the use of IMPROMPTU to support collaboration in other task domains. One particular domain of interest is medical environments. In these environments, there are multiple responsible entities (doctors, nurses, administrators) that use many different information artifacts (lab reports, treatment plans, care notes) as they work towards a shared goal (improving patient care). Often, these workers share common workspaces such as nurses stations, conference rooms, and even patients' rooms. Thus, there is opportunity to leverage an MDE in these workspaces to improve their co-located collaborations. Looking at different domains, such as medical care environments, would allow researchers to better understand how an MDE's features and interfaces would need to changed to better accommodate tasks in domains other than software development.

8.7.3 Integration with Other Communication Tools

The users in our field study indicated a strong preference for combining the functionality of IMPROMPTU's Collaborator Bar with other communication tools such as IM and E-Mail. Combining functionality could significantly benefit users as it would allow them to more easily manage collaborators and more easily transition between different features. For example, an IM conversation could be expanded to included replicated applications.

We are currently exploring designs that would include IM and email functionality within the Collaborator Bar. We are also exploring the ability to provide plug-ins to popular IM

and email clients that would allow users to initiate replications of applications directly within these clients.

Bibliography

1. Apple Macintosh OS X. http://www.apple.com/macosx/.

2. Concurrent Versions System (CVS). http://www.nongnu.org/cvs/.

3. FVWM Window Manager. http://www.fvwm.org/.

4. KDE Window Manager. http://www.kde.org/.

5. Microsoft LiveMeeting. http://office.microsoft.com/livemeeting/.

6. Microsoft Windows Vista. http://www.microsoft.com/windows/.

7. Sawfish Window Manager. http://sawfish.wikia.com/

8. Subversion (SVN). http://subversion.tigris.org/.

9. Abowd, G.D. and Mynatt, E.D. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction, 7* (1). 29-58.

10. Allen, T.J. *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technical Information within R&D Organizations*. MIT Press, Cambridge, MA., 1977.

11. Altman, D.G. *Practical Statistics for Medical Research*. Chapman and Hall, London, England, 1991.

12. Axelrod, R. *The Evolution of Cooperation*. Basic Books, New York, NY, 1984.

13. Backhouse, A. and Drew, P. The Design Implications of Social Interaction in a Workplace Setting. *Environment and Planning B: Planning and Design, 19*. 573-584.

14. Bales, R.F. *Interaction Process Analysis: A Method for the Study of Small Groups*. Addison-Wesley, Cambridge, MA, 1950.

15. Ballagas, R., Ringel, M., Stone, M. and Borchers, J., iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Fort Lauderdale, Florida, 2003), 537-544.

16. Bederson, B.B., Hollan, J.D., Perlin, K., Meyer, J., Bacon, D. and Funas, G.W. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal Visual Languages and Computing*, *7*. 3-31.

17. Benford, S., Bowers, J., Fahlén, L.E., Greenhalgh, C. and Snowdon, D., User Embodiment in Collaborative Virtual Environments. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Denver, CO, USA, 1995), 242-249.

18. Benford, S. and et.al., Designing Storytelling Technologies to Encourage Collaboration Between Young Childred. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (2000), ACM, 556-563.

19. Biehl, J.T. and Bailey, B.P., ARIS: An Interface for Application Relocation in an Interactive Space. in *Proceedings of Graphics Interface*, (London, Ontario, Canada, 2004), A K Peters, 107-116.

20. Biehl, J.T. and Bailey, B.P., Comparing a Textual Interface, Virtual Paths Interface, and Iconic Map Interface for Effective Interaction in an Interactive Workspace. in *Proceedings of the AACE World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA)*, (Orlanado, FL, USA, 2006), Association for the Advancement of Computing in Education.

21. Biehl, J.T. and Bailey, B.P., Improving Interfaces for Managing Applications in Multiple-Device Environments. in *Proceedings of the International Conference on Advanced Visual Interfaces (AVI)*, (Venice, Italy, 2006), ACM, 35-42.

22. Biehl, J.T. and Bailey, B.P., Improving Scalability and Awareness in Iconic Interfaces for Multiple-Device Environments. in *Proceedings of the International Conference on Advanced Visual Interfaces (AVI)*, (Venice, Italy, 2006), ACM, 91-94.

23. Biehl, J.T. and Bailey, B.P., A Toolset for Constructing and Supporting Iconic Interfaces for Interactive Workspaces. in *Proceedings of the Tenth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT)*, (Rome, Italy, 2005), Springer, 699-712.

24. Biehl, J.T., Baker, W.T., Bailey, B.P., Tan, D.S., Inkpen, K. and Czerwinski, M., IMPROMPTU: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and Its Field Evaluation for Co-located Software Development. in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, (Florence, Italy, 2008), ACM, 939-948.

25. Biehl, J.T., Czerwinski, M., Smith, G. and Robertson, G.G., FASTDash: A Visual Dashboard for Fostering Awareness in Software Teams. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (San Jose, CA, USA, 2007), ACM, 1313-1322.

26. Birnholtz, J.P., Gutwin, C., Ramos, G. and Watson, M., OpenMessenger: Gradual Initiation of Interaction for Distributed Workgroups. in *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Florence, Italy, 2008), ACM, 1661-1664.

27. Booth, K.S., Fisher, B.D., Lin, C.J.R. and Argue, R., The "Mighty Mouse" Multi-Screen Collaboration Tool. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (Paris, France, 2002), 209-212.

28. Brinck, T. and Gomez, L.M., A Collaborative Medium for Support of Conversational Props. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, (1992), ACM, 171-178.

29. Brown, A.L. and Palincsar, A.S. Guided, Cooperative Learning and Individual Knowledge Acquisition. in Resenick, L.B. ed. *Knowing, Learning, and Instruction*, Lawrence Erlbaum, New Jersey, 1989, 393-451.

30. Brumitt, B., Meyers, B., Krumm, J., Kern, A. and Shafer, S.A., EasyLiving: Technologies for Intelligent Environments. in *Handheld and Ubiquitous Computing*, (2000), 12-29.

31. Bryant, S., Romero, P. and Boulay, B., Pair Programming and the Re-appropriation of Individual Tools for Collaborative Programming. in *Proceedings of the ACM Conference on Group Interaction (GROUP)*, (2005), 332-333.

32. Chang, B. and Ungar, D., Animation: From Cartoons to the User Interface. in *Proceedings of the ACM Conference on User Interface and Software Technology*, (1993), 45-55.

33. Cheng, L.-T., Hupfer, S., Ross, S. and Patterson, J., Jazzing up Eclipse with collaborative tools. in *Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, (Anaheim, California, 2003), ACM Press, 45-49.

34. Chong, J. and Siino, R., Interruptions on Software Teams: A Comparison of Paired and Solo Programmers. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, (Banff, Alberta, Canada, 2006), ACM, 29-38.

35. Cohen, J. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement, 20*. 37-46.

36. Cottrell, N.B. Social Facilitation. in McClintock, C. ed. *Experimental Social Psychology*, Holt, Rinehart, & Winston, New York, 1972, 183-236.

37. Curtis, B., Krasner, H. and Iscoe, N. A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*, *31* (11). 1268-1287.

38. d'Astous, P., Détienne, F., Robillard, P.N. and Visser, W., Types of dialogs in evaluation meetings: an analysis of technical-review meetings in software development. in *International Conference on the Design of Cooperative Systems (COOP)*, (Cannes, France, 1998), 25-33.

39. Desmond, J.P. Software 500: Deep Concerns Over Security *Software Magazine*, 2006.

40. Dick, A.J. and Zarnett, B., Paired Programming & Personality Traits. in *Proceedings of the 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2002)*, (Sardinia, Italy, 2002).

41. Diehl, M. and Stroebe, W. Productivity Loss in Brainstorming Groups: Towards the Solution of a Riddle. *Journal of Personality and Social Psychology*, *53* (3). 497-509.

42. Dix, A., Finlay, J., Abowd, G. and Beale, R. *Human-Computer Interaction*. Prentice Hall, 2003.

43. Douglas, T. *A Theory of Groupwork Practice*. The Macmillan Press Ltd, Houndsmills, UK, 1993.

44. Elwart-Keys, M., Halonen, D., Horton, M., Kass, R. and Scott, P., User Interface Requirements for Face to Face Groupware. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Seattle, Washington, USA, 1990), ACM, 295-301.

45. Endsley, M.R. Direct Measurement of Situation Awareness: Validity and Use of SAGAT. in Endsley, M.R. and Garland, D.J. eds. *Situation Awareness Analysis and Measurement*, Lawrence Erlbaum Associates, Mahwah, New Jersey, 2000, 147-174.

46. Endsley, M.R., Situation awareness global assessment technique (SAGAT). in *IEEE National Aerospace and Electronics Conference*, (Dayton, OH, 1988), IEEE, 789-795.

47. Everitt, K., Shen, C., Ryall, K. and Forlines, C., MultiSpace: Enabling Electronic Document Micro-Mobility in Table-Centric, Multi-Device Environments. in *Proceedings of the IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop)*, (Adelaide, Australia, 2006), IEEE Computer Society, 27-34.

48. Forsyth, D.R. *Group Dynamics*. Wadsworth Publishing Company, Belmont, CA, 1999.

49. Froehlich, J. and Dourish, P., Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams. in *Proceedings of the International Conference on Software Engineering*, (2004), 387-396.

50. Gabarro, J.J. The Development of Working Relationships. in Lorsch, J.W. ed. *Handbook of Organizational Behavior*, Prentice-Hal, Englewood Cliffs, NJ, 1987, 172-189.

51. Gabora, L., Cognitive Mechanisms Underlying the Creative Process. in *Proceedings of the ACM Conference on Creativity & Cognition*, (Loughborough, UK, 2002), ACM, 126-133.

52. Garfinkel, D., Welti, B.C. and Yip, T.W. HP SharedX: A Tool for Real-Time Collaboration. *Hewlett-Packard Journal, April 2004*. 23-36.

53. Geen, R.G. Evaluation Apprehension and the Social Facilitation/Inhibition of Learning. *Motivation and Emotion, 7* (2). 203-212.

54. Glass, R.L. Extreme programming: the good, the bad, and the bottom line. *IEEE Software, 18* (6). 111-112.

55. Grudin, J. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM, 37* (1). 92-105.

56. Gutwin, C., Fedak, C., Watson, M., Dyck, J. and Bell, T., Improving Network Efficiency in Real-Time Groupware with General Message Compression. in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, (Banff, Alberta, Canada, 2006), ACM, 119-128.

57. Gutwin, C. and Greenberg, S. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Journal of Computer-Supported Cooperative Work* (3-4). 411-446.

58. Gutwin, C. and Greenberg, S., The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. in *IEEE 9th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, (2000).

59. Gutwin, C., Penner, R. and Schneider, K., Group Awareness in Distributed Software Development. in *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, (Chicago, Illinois, USA, 2004), ACM, 72-81.

60. Ha, V., Inkpen, K.M., Mandryk, R.L. and Whalen, T., Direct Intentions: The Effects of Input Devices on Collaboration around a Tabletop Display. in *Proceedings of the IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop)*, (Adelaide, Australia, 2006), IEEE Computer Society, 177- 184.

61. Hailpern, J., Hinterbichler, E., Leppert, C., Cook, D. and Bailey, B.P., TEAM STORM: Demonstrating an Interaction Model for Working with Multiple Ideas During Creative Group Work. in *Proceedings of the ACM Conference on Creativity & Cognition*, (Washington, DC, USA, 2007), ACM, 193-202.

62. Hart, S.G. and Stateland, L.E. Development of NASA-TLX (Task Load Index): Results of emperical and theoretical research. in Hancock, P.A. and Meshkati, N. eds. *Human Mental Workload*, North-Holland, Amsterdam, 1988, 139-183.

63. Heiberg, S., Puus, U., Salumaa, P. and Seeba, A., Pairprogramming effect on developers productivity. in *Proceedings of the thd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2003)*, (Genova, Italy, 2003).

64. Henderson, A. and Card, S.K. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-based Graphical User Interface. *ACM Transactions on Graphics, 5* (3). 211-243.

65. Herbsleb, J.D. and Mockus, A. An Empirical Study of Speed and Communication in Globally-Distributed Software Development. *IEEE Transactions of Software Engineering, 29* (6). 481- 494.

66. Hewett, T.T. Informing the Design of Computer-Based Environments to Support Creativity. *International Journal of Human-Computer Studies, 63* (4-5). 383-409.

67. Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P. and Smith, M., Stitching: Pen Gestures that Span Multiple Displays. in *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, (Gallipoli, Italy, 2004), ACM, 23-31.

68. Izadi, S., Brignull, H., Rodden, T., Rogers, Y. and Underwood, M., Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media. in *Proceedings of ACM Symposium on User Interface Software and Technology*, (Vancouver, BC, Canada, 2003), ACM, 159-168.

69. Janis, I.L. *Victims of Groupthink*. Houghton Mifflin Company, Boston, 1972.

70. Jiang, T.M. and Sankaran, L. Fast, Portable Application Mirroring. *IEEE Software, 12* (2). 57-63.

71. Johansen, R., Charles, J., Mittman, R. and Saffo, P. *Groupware: Computer Support for Business Teams* Free Press, 1988.

72. Johanson, B. and Fox, A., The Event Heap: A Coordination Infrastructure for Interactive Workspaces. in *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, (2002), 83-93.

73. Johanson, B., Fox, A. and Winograd, T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing*, *1* (2). 67-74.

74. Johanson, B., Hutchins, G., Winograd, T. and Stone, M., PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (Paris, France, 2002), ACM, 227-234.

75. Johanson, B., Ponnekanti, S., Sengupta, C. and Fox, A., Multibrowsing: Moving Web Content Across Multiple Displays. in *Proceedings of Ubicomp*, (2001), 346-353.

76. Kahn, A., Fitzmaurice, G., Almeida, D., Burtnyk, N. and Kurtenbach, G., A Remote Control Interface for Large Displays. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (Santa Fe, NM, USA, 2004), ACM Press, 127-136.

77. Karau, S. and Williams, K.D. Social Loafing: A Meta-Analytic Review and Theoretical Integration. *Journal of Personality and Social Psychology*, *65* (4). 681-706.

78. Katz, R. and Tushman, M. Communication Patterns, Project Performance, and Task Characteristics: an Empirical Evaluation and Integration In an R&D Setting. *Organizational Behavior and Human Performance*, *23*. 139-162.

79. Krauss, R.M. and Fussel, S.R. Mutual Knowledge and Communicative Effectiveness. in Galegher, J., Kraut, R. and Egido, C. eds. *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, Lawrence Erlbaum Associates Inc., Hillsdale, N.J., 1990, 111-145.

80. Kraut, R.E. and Streeter, L.A. Coordination in Software Development. *Communications of the ACM*, *38* (3). 69-81.

81. Latané, B., Williams, K. and Harkins, S. Many Hands Make Light the Work: The Causes and Consequences of Social Loafing. *Journal of Personality and Social Psychology*, *37* (6). 822-832.

82. LaToza, T.D., Venolia, G. and DeLine, R., Maintaining Mental Models: A Study of Developer Work Habits. in *Proceedings of the Internation Conference on Software Engineering (ICSE)*, (Shanghai, China, 2006), ACM, 492-501.

83. Layman, L., Williams, L. and Cunningham, L., Exploring Extreme Programming in Context: An Industrial Case Study. in *Proceedings of the IEEE Agile Development Conference*, (2004), IEEE, 32- 41.

84. Lewis, C. and Rieman, J. Task-Centered User Interface Design, 1994.

85. Lewis, J.R. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction, 7* (1). 57-78.

86. Lugt, R. Brainsketching and How It Differs from Brainstorming. *Creativity and Innovation Management, 11* (1). 43-54.

87. Lui, K.M. and Chan, K.C.C., When does a pair outperform two individuals. in *Proceedings of the thd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2003),* (Genova, Italy, 2003), Springer.

88. MacIntyre, B., Mynatt, E.D., Voida, S., Hansen, K.M., Tullio, J. and Corso, G.M., Support for Multitasking and Background Awareness Using Interactive Peripheral Displays. in *Proceedings of ACM Symposium on User Interface Software and Technology*, (Orlando, Florida, USA, 2001), ACM, 41-50.

89. Mack, R.L. and Nielsen, J. *Usability Inspection Methods: Executive Summary*. Morgan Kaufmann, San Francisco, CA, 1995.

90. Mandviwalla, M. and Olfman, L. What do groups need? A Proposed Set of Generic Groupware Requirements. *ACM Transactions on Computer-Human Interaction, 1* (3). 245-268.

91. MCI. Meetings in America. MCI ed., 1998, 21.

92. Miller, G. The Magical Number Seven, Plus or Minus Two. *The Pscychological Review, 63*. 81-97.

93. Morris, M.R., Paepcke, A., Winograd, T. and Stamberger, J., TeamTag: Exploring Centralized verse Replicated Controls for Co-located Tabletop Groupware. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (2006), ACM, 1273-1282.

94. Morris, M.R., Ryall, K., Shen, C., Forlines, C. and Vernier, F., Beyond "Social Protocols": Multi-User Coordination Policies for Co-located Groupware. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work,* (Chicago, Illinois, USA, 2004), ACM Press, 262-265.

95. Myers, B.A., Stiel, H. and Gargiulo, R., Collaboration Using Multiple PDAs Connected to a PC. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work,* (Seattle, WA, 1998), 285-294.

96. Nacenta, M.A., Aliakseyeu, D., Subramanian, S. and Gutwin, C., A Comparison of Techniques for Multi-Display Reaching. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Portland, Oregon, 2005), 371-380.

97. Nacenta, M.A., Sakurai, S., Yamaguchi, T., Miki, Y., Itoh, Y., Kitamura, Y., Subramanian, S. and Gutwin, C., E-conic: a Perspective-Aware Interface for Multi-Display Environments. in *Proceedings of the Symposium on User Interface Software and Technology (UIST)*, (Newport, Rhode Island, USA, 2007), ACM, 279 - 288.

98. Nosek, J.T. The case for collaborative programming. *Communications of the ACM, 41* (3). 105-108.

99. Olson, G.M., Olson, J.S., Carter, M.R. and Storrøsten, M. Small Group Design Meetings: An Analysis of Collaboration. *Human-Computer Interaction, 7* (4). 347-374.

100. Parrish, A., Smith, R., Hale, D. and Hale, J. A Field Study of Developer Pairs: Productivity Impacts and Implications. *IEEE Software, 21* (5). 76-79.

101. Paulus, P.B. and Yang, H.-C. Idea Generation in Groups: A Basis for Creativity in Organizations. *Organizational Behavior and Human Decision Processes, 82* (1). 76-87.

102. Penn, A., Desyllas, J. and Vaughan, I. The Space of Innovation. *Environment and Planning B: Planning and Design, 26*. 193-218.

103. Perlow, L.A. The Time Famine: Towards a Sociology of Work Time. *Administrative Science Quarterly, 44* (1). 57-81.

104. Ponnekanti, S.R., Lee, B., Fox, A., Hanrahan, P. and Winograd, T., iCrafter: A Service Framework for Ubiquitous Computing Environments. in *Proceedings of the Conference on Ubiquitous Computing Conference (UBICOMP)*, (2001), 56-75.

105. Reder, S. and Schwab, R.G., The Temporal Structure of Cooperative Activity. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, (1990), ACM, 303-320.

106. Rekimoto, J., Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. in *ACM Symposium on User Interface Software and Technology*, (Banff, Albera, Canada, 1997), ACM, 31-39.

107. Rekimoto, J., Time-Machine Computing: A Time-centric Approach for the Information Environment. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (1999), ACM, 45-54.

108. Rekimoto, J. and Saitoh, M., Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Pittsburgh, PA, USA, 1999), ACM, 378-385.

109. Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D. and Dantzich, M.v., Data Mountain: Using Spatial Memory for Document Management. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (Sanfrancisco, CA, USA, 1998), ACM, 153-162.

110. Robertson, G., Dantzich, M.v., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. and Gorokhovsky, V., The Task Gallery: A 3D Window Manager. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (2000), ACM, 494-501.

111. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D. and Smith, G., Scalable Fabric: Flexible Task Management. in *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI)*, (Gallipoli, Italy, 2004), ACM, 85-89.

112. Robillard, P.N. and Robillard, M.P. Types of Collaborative Work in Software Engineering. *Journal of Systems and Software*, *53* (2). 219-224.

113. Rogers, Y., Lim, Y.-K. and Hazlewood, W.R., Extending Tabletops to Support Flexible Collaborative Interactions. in *Proceedings of the IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop)*, (Adelaide, Australia, 2006), IEEE Computer Society, 71-78.

114. Rogers, Y. and Lindley, S. Collaborating Around Vertical and Horizontal Displays: Which Way is Best? *Interacting with Computers*, *16*. 1133-1152.

115. Román, M. and Campbell, R., Providing Middleware Support for Active Space Applications. in *ACM/IFIP/USENIX International Middleware Conference*, (2003).

116. Román, M., Hess, C., Cerqueira, R., Ranganat, A., Campbell, R. and Nahrstedt, K. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, *1* (4). 74-83.

117. Román, M., Ho, H. and Campbell, R., Application Mobility in Active Spaces. in *International Conference on Mobile and Ubiquitous Multimedia*, (2002).

118. Ryall, K., Forlines, C., Shen, C. and Morris, M.R., Exploring the Effects of Group Size and Table Size on Interactions with Tabletop Shared-Display groupware. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, (Chicago, IL, USA, 2004), ACM, 284-293.

119. Sarma, A., Noroozi, Z. and Hoek, A.v.d., Palantír: Raising Awareness among Configuration Management Workspaces. in *Proceedings of the Internation Conference on Software Engineering (ICSE)*, (Portland, Oregon, USA, 2003), IEEE Computer Society, 444-454.

120. Schilit, B.N., Adams, N.I. and Want, R., Context-Aware Computing Applications. in *Proceedings of the Workshop on Mobile Computing Systems and Applications*, (1994), IEEE Computer Society, 85-90.

121. Schilit, B.N. and al., e., Customizing Mobile Applications. in *USENIX Symposium on Mobile and Location-Independent Computing*, (1993), 129-138.

122. Schön, D.A. *Reflective Practitioner: How Professionals Think in Action*. Basic Books, 1990.

123. Schwaber, K. and Beedle, M. *Agile Software Development with SCRUM*. Prentice Hall, Upper Saddle River, NJ, 2002.

124. Scott, S.D., Grant, K.D. and Mandryk, R.L., System Guidelines for Co-located Collaborative Work on a Tabletop Display. in *Proceedings of the European Conference on Computer Supported Cooperative Work*, (Helsinki, Finland, 2003), Kluwer Academic Publishers, 159-178.

125. Segal, L.D. Designing Team Workstations: The Choreography of Teamwork. in Hancock, P.A. ed. *Local Applications of the Ecological Approach to Human-Machine Systems*, Lawrence Erlbaum Associates Inc., Hillsdale, N.J., 1995, 392-415.

126. Sharp, H. and Robinson, H. An Ethnographic Study of XP Practice. *Empirical Software Engineering*, *9*. 353-375.

127. Shen, C., Everitt, K.M. and Ryall, K., UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. in *Proceedings of the Conference on Ubiquitous Computing Conference (UBICOMP)*, (2003), 281 - 288.

128. Shen, C., Vernier, F.D., Forlines, C. and Ringel, M., DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (2004), 167-174.

129. Singer, J., Practices of Software Maintenance. in *Proceedings of the International Conference on Software Maintenance*, (Washington D.C., USA, 1998), 139-145.

130. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., Horvitz, E. and Andrews, D., GroupBar: The TaskBar Evolved. in *The Australasian Computer-Human Interaction Conference (OZCHI)*, (2003), 34-43.

131. Sousa, J.P. and Garlan, D., Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. in *IEEE Conference on Software Architecture*, (2002), 29-43.

132. Souza, C., Redmiles, D. and Dourish, P., "Breaking the code", moving between private and public work in collaborative software development. in *Proceedings of the ACM Conference on Supporting Group Work*, (2003), 105-114.

133. Stefik, M., Bobrow, D.G., Foster, G., Lanning, S. and Tatar, D. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Transactions on Information Systems, 5* (2). 147-167.

134. Stefik, M., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S. and Suchman, L. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Communications of the ACM, 30* (1). 32-47.

135. Stewart, J., Bederson, B.B. and Druin, A., Single Display Groupware: A Model for Co-present Collaboration. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Pittsburgh, PA, 1999), 286-293.

136. Streitz, N.A., Giessler, J., Holmer, T., Konomi, S., Muller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P. and Steinmetz, R., i-LAND: AN Interactive Landscape for Creativity and Innovation. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (1999), 120-127.

137. Streitz, N.A., Rexroth, P. and Holmer, T., Does Roomware Matter? Investigating the Role of Personal and Public Information Devices and their Combination in Meeting Room Collaboration. in *Proceedings of E-CSCW*, (1997), 297-312.

138. Tan, D.S., Meyers, B. and Czerwinski, M., WinCuts: manipulating arbitrary window regions for more effective use of screen space. in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, (Vienna, Austria, 2004), ACM Press, 1525-1528.

139. Tandler, P., Prante, T., Muller-Tomfelde, C., Streitz, N. and Steinmetz, R., ConnecTables: dynamic coupling of displays for the flexible creation of shared workspaces. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (Orlando, Florida, 2001), 11-19.

140. Tang, J.C. Findings from Observational Studies of Collaborative Work. *International Journal of Man-Machine Studies, 34*. 143-160.

141. Tee, K., Greenberg, S. and Gutwin, C., Providing Artifact Awareness to a Distributed Group through Screen Sharing. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, (Banf, Calgary, Canada, 2006), ACM, 99-108.

142. Totten, S., Sills, T., Digby, A. and Russ, P. *Cooperative Learning: A Guide to Research*. Garland, New York, 1991.

143. Tsai, W. and Ghoshal, S. Social Capital and Value Creation: the Role of Infirm Networks. *Academy of Management Journal of Personality and Social Psychology*, *41* (4). 464-476.

144. Tse, E. and Greenberg, S., Rapidly Prototyping Single Display Groupware through the SDGToolkit. in *Proceedings of the Fifth Australian User Interface Conference*, (2004), 101-110.

145. Weiser, M. The Computer of the Twenty-First Century *Scientific American*, 1991, 94-104.

146. Wickens, C.D. Multiple Resources and Performance Prediction. *Theoretical Issues in Ergonomic Science*, *3* (2). 159-177.

147. Wigdor, D., Shen, C., Forlines, C. and Balakrishnan, R., Table-Centric Interactive Spaces for Real-Time Collaboration. in *Proceedings of the International Conference on Advanced Visual Interfaces (AVI)*, (Venice, Italy, 2006), ACM, 103-107.

148. Williams, L. and Kessler, R. All I Really Need To Know about Pair Programming I Learned in Kindergarten. *Communications of the ACM*, *43* (5). 108-114.

149. Williams, L., Kessler, R., Cunningham, W. and Jeffries, R. Strengthening the case for pair programming. *IEEE Software*, *17* (4). 19-25.

150. Wu, J., Graham, T.C.N. and Smith, P.W., A Study of Collaboration in Software Design. in *Proceedings of the International Symposium on Empirical Software Engineering (ISESE)*, (2003), IEEE Computer Society.

151. Xia, S., Sun, D., Sun, C., Chen, D. and Shen, H., Leveraging single-user applications for multi-user collaboration: the CoWord approach. in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, (2004), 162-171.

Author's Biography

Jacob ("Jake") Biehl received both his M.S. in Computer Science and B.S. in Statistics and Computer Science from the University of Illinois at Urbana-Champaign in 2004 and 2002, respectively. His dissertation work focuses on the design, development, and evaluation of interfaces, interaction techniques, and interaction frameworks to support natural collaborative practices in multiple-display environments. More broadly, his research interests span many dimensions of computer-supported cooperative work, including development of systems and visualizations to support group awareness, and tools that improve communication and coordination within groups that are geographically separated and culturally divergent. Jake was the recipient of the Verizon Fellowship in 2002-2003. His work on IMPROMPTU was awarded first place in the 2007 Siebel Computing Habitat Competition. He will be joining Fuji Xerox Palo Alto Laboratory (FXPAL) in California as a full-time Research Scientist in August 2008.